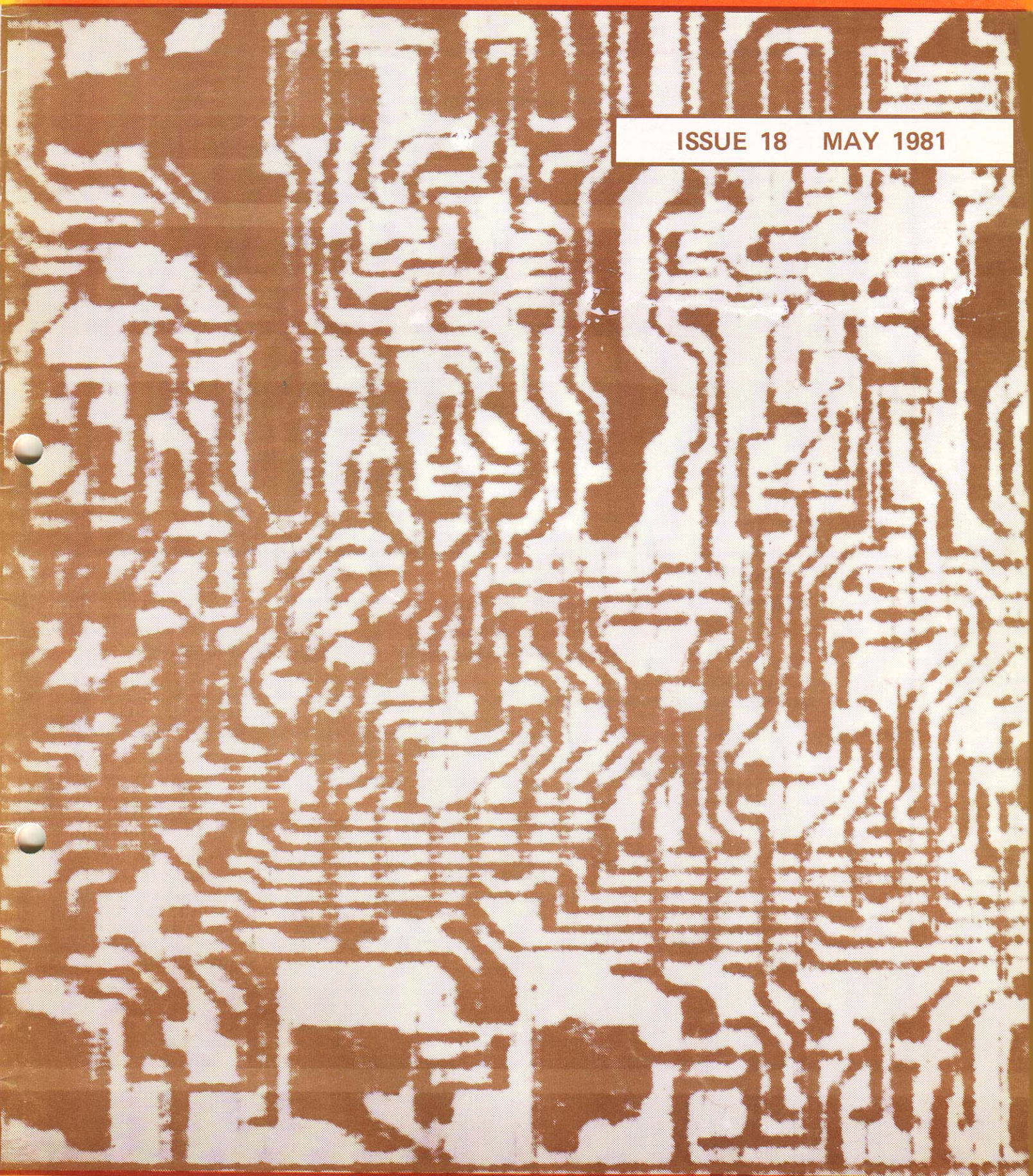
TRS-80 SYSTEM 80 VIDEO GENIE



ISSUE 18 MAY 1981

MICRO-80

P.O. BOX 213, GOODWOOD, S.A., 5034, AUSTRALIA. TELEPHONE (08) 272 0966. PRICE; AUS. $2.50, NZ. 3.00, U.K. £1.50

***** ABOUT MICRO-80 *****

EDITOR:	IAN VAGG
SOFTWARE EDITOR:	CHARLIE BARTLETT
HARDWARE EDITOR:	EDWIN PAAY
U.K. CORRESPONDENT:	TONY EDWARDS

MICRO-80 is an international magazine devoted entirely to the Tandy TRS-80 microcomputer and the Dick Smith System 80/Video Genie. It is available at the following prices (all prices shown in Aus.$ except for U.K. prices which are in pounds Sterling).

12 months subscription	Aus.	$24.00
	NZ.	$36.00 (Airmail)
	Hong Kong	$46.00 (Airmail)
	U.K.	£16.00
Single Copy	Aus.	$2.50
	N.Z.	$3.50 (Airmail)
	Hong Kong	$4.25 (Airmail)
	U.K.	£1.50
Months programs on cassette	Aus.	$3.50
	N.Z.	$4.00 (Airmail)
	Hong Kong	$4.50 (Airmail)
(at present available from Australia only)	U.K.	$4.75 (Airmail)
12 months subscription to magazine and cassette	Aus.	$60.00
	N.Z.	$78.00 (Airmail)
	Hong Kong	$88.00 (Airmail)
	U.K.	£41.00 (Airmail)

Special bulk purchase rates are also available to computer shops etc. Please use the form in this issue to order your copy or subscription.

The purpose of MICRO-80 is to publish software and other information to help you get the most from your TRS-80, System 80 or Video Genie and their peripherals. MICRO-80 is in no way connected with either the Tandy or Dick Smith organisations.

** WE WILL PAY YOU TO PUBLISH YOUR PROGRAMS **
Most of the information we publish is provided by our readers, to whom we pay royalties. An application form containing full details of how you can use your TRS-80 or System 80 to earn some extra income is included in every issue.

** CONTENT **
Each month we publish at least one applications program in Level I BASIC, one in Level II BASIC and one in DISK BASIC (or disk compatible Level II). We also publish Utility programs in Level II BASIC and Machine Language. At least every second issue has an article on hardware modifications or a constructional article for a useful peripheral. In addition, we run articles on programming techniques both in Assembly Language and BASIC and we print letters to the Editor and new product reviews.

** COPYRIGHT **
All the material published in this magazine is under copyright. That means that you must not copy it, except for your own use. This applies to photocopying the magazine itself or making copies of programs on tape or disk.

** LIABILITY **
The programs and other articles in MICRO-80 are published in good faith and we do our utmost to ensure that they function as described. However, no liability can be accepted for the failure of any program or other article to function satisfactorily or for any consequential damages arising from their use for any purpose whatsoever.

***** CONTENTS *****

MICRO-80 is registered by Australia Post — Publication SQB2207 Category B

AUSTRALIAN OFFICE AND EDITOR:

MICRO-80, P.O. BOX 213, GOODWOOD, SOUTH AUSTRALIA, 5034. TEL. (08) 211 7244

U.K. SUBSCRIPTION DEPT:

24 WOODHILL PARK, PEMBURY, TUNBRIDGE WELLS, KENT TN2 4NW.

Printed by:

Shovel & Bull Printers, 312A Unley Road, HYDE PARK, S.A. 5061.

Published in Australia by MICRO-80, 433 Morphett Street, ADELAIDE.

★ ★ ★ FREE SOFTWARE OFFER ★ ★ ★

EVERY NEW AUSTRALASIAN SUBSCRIBER TO MICRO-80 WILL RECEIVE A FREE CASSETTE CONTAINING THREE LEVEL I AND THREE LEVEL II PROGRAMS PLUS COMPREHENSIVE DOCUMENTATION. THE RETAIL VALUE OF THE SOFTWARE WOULD EXCEED THE COST OF THE SUBSCRIPTION!!!

***** EDITORIAL *****

The change to double-width mode for our program listings last issue evinced a lot of response from readers, mostly negative. As a result, we have reverted to the previous format but with one significant difference. This month we have used the ever versatile EPSOM MX-80 in its emphasised print mode, which reproduces much denser characters. We believe this should achieve our aim of legible listings which are easy to read. If you have any comments to make on this month's listings, please do not hesitate to write in.

We are proud of the quality of the programs we publish in MICRO-80, not only because of their performance but because of the accuracy of the listings. Many readers have commented how they can always rely on our listings and have compared them favourably with many other magazines. This is no accident. Before a program is published, it has been thoroughly tested by the appropriate Software Editor. In many cases the author is asked to make minor modifications and sometimes the Software Editors themselves rewrite sections of the programs. It can take several months from the time a program is submitted until it is actually published. Once the programs for a particular issue are debugged, the master cassette is punched out, a backup copy on disk is made and the programs are LLISTed. For a long time this proved to be a pretty foolproof system. However, recently we have been disturbed to find a number of errors creeping into the listings and onto the cassettes. On investigation, we discovered that the increased incidence of errors coincided with a change in our process several issues ago. Then, in response to requests from finger-weary readers, we commenced renumbering all Level II programs to facilitate the use of AUTO entering. The renumbering was carried out after the original programs had been carefully checked out and just before the master cassette was punched. Apparently, during the renumbering process, spurious line numbers were occasionally generated after GOTO's etc. The obvious culprit, the renumbering routine, turned out to be an innocent victim of circumstances. The actual fault was eventually found to be a noisy expansion interface causing random changes in data held in memory. The fault has now been rectified and our listings should revert to their usual high standard. In the meantime, as errors are reported to us, we will correct them in MICRO-BUGS. One of the first programs to be affected was Mighty Mormar. In debugging this program, Charlie Bartlett made some improvements. We will publish the revised version of Mighty Mormar in a future issue. We apologise for inconvenience caused to our readers by this fault.

On a much happier note, we are pleased to announce the appointment of Tony Edwards as our first U.K. correspondent and European Software Coordinator. Elsewhere in this issue, you will find Tony's first article in which he introduces himself and explains the procedure to be followed by our European readers when submitting programs for publication. You may also send articles for publication to Tony who will pass them on to Australia. Here at MICRO-80, we are committed to serving '80 users all over the world and we are sure that, with Tony's appointment, we will be able to improve the service to our U.K. and European readers still further.

A less happy announcement is that Michael Svensdotter, our Level I Software Editor, has resigned due to lack of time occasioned by a change in employment. Michael is one of the most experienced Level I programmers we know and will be a loss to us all. We wish Michael well in his new job and look forward to the occasional contribution from him as time permits.

Charlie Bartlett, our Level II Software Editor, who is also no mean hand at Level I programming, has assumed overall responsibility for editing of all MICRO-80 software. It so happens that our stocks of Level I software for publication are a little low so, if you have any programs you think we might like to publish, please send them in to Charlie.

- 0000000000 -

***** MICRO-BUGS *****

In which we correct those errors which seem to creep in, no matter how careful we are.

FINANCE & REMOVING LINEAR BRACKETS - Issue 16 Pages 22 and 23.

It finally happened. As we were assembling the magazine, we mixed up two listings. On page 22, line 960 and all lines below it, should go at the base of page 23. The FINANCE program ends at line 930. REMOVING LINEAR BRACKETS ends at line 1220.

VICTORY AT BATHURST - Issue 16 Pages 32 and 33.

Page 32, Line 2930:-

```
GOTO 8100  should be  GOTO 2900
```

Page 33, Line 3690:-

```
GOTO 8985  should be  GOTO 3760
```

Page 33, Line 3890 should read:-

```
3890 CLS:PRINT@330, "LAP 163 HAS BEEN COMPLETED!!!"
```

- 0000000000 -

***** BETTER BASIC PROGRAMMING - by Rod Stevenson *****

** INTRODUCTION **

This is the first instalment of a proposed series which is intended as a companion to the now-expired "GT BASIC".

It will be modelled on the same format used for similar talks to the Adelaide User's Group - the wide acceptance of which was the reason for your Esteemed Editor to request such a series. The writer is by no means an 'expert', so hopefully the phraseology will be acceptable to most; stimulating to the beginner, providing the seasoned computerist with some insights and points forgotten from earlier days. Unlike a lot of 'beginners' treatises, we do expect some knowledge. If you are convinced the instruction manual is too hard for you to discover how to LIST (yes, this is a real example from our experience), then this is not for you. Because of the format we are using, we can only assume some knowledge on subjects not currently the one under review. An example: while dealing with strings we will mention arrays with no more explanation than needed for the case-in-point. However, when we come to the subject of arrays, we will cover all. Perhaps not all - the deeper complexities of some subjects will be the subject of a further composite article when we have led the self-confessed 'beginners' to realise that it is not all a mystery, and they can have confidence to pursue their own investigations of whatever takes their fancy!

When allied thoughts occur, we will wander off (ramble?) the subject.

In fact, we strongly emphasise the only path to true understanding is to put our words into practice. By so-doing, you will get a 'real feel' for the subject. Every one of the more experienced users in the Adelaide Group has no reluctance in admitting this has been the way to success. Like the old story says, you only get out what you put in!

Subjects we propose to treat:

1. Strings
2. Arrays
3. Input checking
4. Formatting output
5. Machine language in BASIC
6. Sound routines
7. More advanced points of any of above
8. Points requested by readers.

The episode on machine language will be strictly-for-beginners - and this, of necessity, because your writer knows little enough. It, too, will follow the format established in the Adelaide Users' Group Newsletter. The aim is to show that machine language is not something to be put aside till later, or discarded as too hard. In fact, some things can only be done in machine language (e.g. modifications to the operating system); some things need machine language for speed. To whet your appetite :

```
POKE 16396,151: POKE 16397,201
```

The same effect will be accomplished by:

```
POKE 16396,175: POKE 16397,201
```

So too will:

```
POKE 16396,23
```

but we will tell you later why the former are better. Now that you have done this...we'll wait for you...we are sure you must agree with us that this machine language routine to disable BREAK was not hard. And that's the level we'll be staying at - but with full explanations...NO, this paragraph wasn't written by Eddy Paay!

Except in a few instances, we will be catering solely for non-disk users. Most of what we say will apply to disk, but we will not be indulging in any disk-only items. The real reason for this is that the writer is without disk. But also, we assume that disk-people will have the knowledge necessary to extend our comments and routines to their particular cases. For those curious about DISK BASIC, we can recommend "LEVEL 3 BASIC" by Microsoft - though you'll probably need a real use to justify the price of $59.95. Not that the price is exhorbitant for the 5½K of extended BASIC provided.

Those who wonder why we keep mentioning the Adelaide User's Group can satisfy their curiosity by a $5 newsletter subscription, 60c. for some back issues, 28c. for current issue. Just send to the address in the Groups' page.

Above all, we are not writing for our own amusement. We demand feedback from readers. If you want a special subject covered, you need only ask. If you don't approve of the format, tell us, and we'll change if it's a general wish. If you have a better way, tell us so we can spread your wisdom. Don't harbour uncharitable thoughts until you've at least given us the chance to redeem ourselves.

** STRINGS **

WHAT IS ONE

A string consists of characters contained within quotes. It can allegedly be up to 255 characters long, but unless you have a definite need, don't. While on that subject, we suggest not having long and involved program lines - if you do want to conserve space, write your program normally with reasonable length lines, then use a program such as "PACKER" to compress it to approximately 80% of its former length.

QUOTES

We must admit there is room for confusion as to whether a string needs to be enclosed in quotes. The rule is, if in doubt, do. When assigning a variable there must always be quotes - as A$ = "QUOTES". However, as an input or a data item, quotes are needed only if there's a delimiter (colon, comma, leading blank) in the string.

We know most times you can get away with leaving off closing quotes. But we suggest you don't, as herein lies a multitude of traps for the unwary. The most obvious one is the loss of control of line-feed, which may not show up till later in the program. Then too there's the subtle horrific trap which happens with RENUMBER or PACKER. Yes, we know the experienced will scoff at our caution, but you've been warned.

While thinking of laziness, we're moved to implore you not to leave THEN out of IF..... THEN, or to use its alleged abbreviation of a comma. This will also cause problems which may not immediately reveal themselves such as with ELSE, GOSUB, and with utility programs, we know it's easy enough to think "it won't happen to me"; when it does, you won't be the first to have had the same thought.

DEFINING

Strings need space in which to work. The CLEAR statement is used to clear an area in memory solely for this purpose. Generally, you should clear the length of strings plus double the length of the longest one. CLEAR will also re-initialise any variables assigned, destroy DIM & DEF statements and generally do as its name suggests and CLEAR a lot of things you probably didn't want cleared.

So use CLEAR as one of the first things at the start of your program. If you don't CLEAR, BASIC will allocate 50.

Having CLEARed, you can now DEFine variables as strings, to save having to put $ after the variable every time it's used. However, the trap is that every variable starting with that name will be a string, e.g. if DEFSTRA, A will become a string variable, but so too will A1, AQ, A2. Actually the writer doesn't generally use the DEFSTR because of the real likelihood of forgetting it's been done.

And thinking of variables - we suggest using meaningful names. It costs nothing and could save a degree of mental exertion at a later date (next day?). We also suggest looking in the manual for the explanation of defining variables as integer, double precision, single precision. We don't propose to go into it at all but, of course, if there's an overwhelming demand from readers...

Again, the writer doesn't use a global DEFINT for fear of forgetting having done so.

It is still necessary to DIM string arrays over 10, but as arrays are the proposed subject of the next instalment, we'll not get side-tracked here.

OPERATORS

Arithmetic operators apply to strings, but only += <>. One could hardly expect to be able to cube a string!

There are times, though, when it would be handy to be able to subtract a letter from a string and one can't just use - (minus). While this subject will be enlarged upon in the episode on input checking, briefly, the method is:

```
I $ = LEFT$ (I$, LEN (I$) - 1)
```

Similar manipulations can be applied to search for particular letters, etc. The comparison for = proceed from left to right until a mis-match is found. The ASCII characters are compared one by one, so by consulting a table of ASCII values, one can determine what BASIC will consider is ≠ or = to what.

The plus sign will simply add (called concatenate) two strings to make a string equal to the sum of the length of the former separate strings. It will not add in the arithmetic sense, which is why it is called concatenating, e.g. A$ = "WER": B$ = "ASD": C$ = A$ + B$ will give C$ = "WERASD".

The plus sign can also be used to print strings, without concatenating them previously, e.g. PRINT A$ + B$, will give the same result as PRINT A$; B$.

FUNCTIONS

This whole list can be found in the Manual. The only reason for including it here is to avoid multitudes of incensed readers complaining that it has been left out.

ASC (A$) returns decimal ASCII value of first character of A$; useful for determining the ASCII value without having to look up a table, such as shift characters, ↓ , etc. can be as PRINT ASC ("!").

CHR$ (X) returns the character which is represented by the ASCII value X; useful for printing a normally unprintable character which would otherwise be a delimiter in a string, such as quotes, e.g.

PR$ = "ASD" + CHR$ (34) + "QWE" + CHR$ (34) will assign PR$ to be ASD "QWE", which would not otherwise be possible. (Opposite of ASC (A$)

FRE(A$) will show how much string space (allocated by CLEAR) is left. The A$ is a dummy variable and does not mean the space has been allocated to A$.

LEN(A$) gives the length (number of characters) in A$ and is vital in manipulations and testing string input.

STR$(A) converts the numeric value of a variable A to a string variable. This allows it to be worked on (in a non-numeric sense) by the string manipulator functions. E.g., to reverse a number represented by I

I$ = STR$(I)

Then I$ = MID$(I$,3,1) + MID$(I$,2,1) + MID$(I$,1,1). It inserts a space before the number which is normally present in a numeric variable to allow for a + or a - sign.

VAL(A$) converts a string number to a numeric value so it can be worked on arithmetically. If there are alpha characters in the string only the preceding numbers will be recognised. Returns a double precision value. VAL and TR perform complimentary functions. They are important as they make strings the most powerful and useful functions in our version of BASIC.

NOTE: ASC(A$) and VAL(A$) are not the same. STRING$ (n,c) gives a string of length n, composed of characters "c". It is useful for quickly printing strings. C can be an ASCII value or any character enclosed by quotes. STRING$ need not be assigned to a variable; you can PRINT STRING$ (n,c) for formatted output.

A$ (n,m) is a string function in other BASICs which should be understood to be able to convert programs written in other than Microsoft BASIC. Although it appears to be a two dimensional string, it is actually a substring of A$, commencing at character n from the start, ending at character m from start.

LEFT$(A$,m) gives a substring of A$, starting at the left, and of m characters in length. Particularly useful in getting input such as YES/NO, which may be input as Y/N. E.g. If LEFT$(A$,1)><"Y" THEN...

RIGHT$(A$,n) gives a substring of A$, starting n characters from the left and containing all the remaining characters in A$. It can be used after STR$ to get rid of the unwanted space that was present in the numeric variable. It can also be used to ensure that the expected number of characters has been input by the operator. More of this in the instalment concerning checking input, but briefly, if entering a date, to allow for either single or double digit entry, i.e. 06 or 6, use:

I$ = "0" + I$: I$ = RIGHT$ (I$,2)

RIGHT$ is also useful for picking the postcode from an address, for sorting, etc.

MID$(A$,n,m) returns a substring of A$, starting n characters from left, of m characters in length. This is probably the most useful of the manipulation functions. It allows you to pick off part of a string and match it against another. The Disk BASIC function of INSTRING can be simulated. However, its real value lies in input checking. Some (your author included) use strings instead of arrays, and it is MID$ which allows this. More about this in the more advanced instalment.

STRING PACKING

We thought this deserved a mention here, although it is more advanced than the rest of this instalment. The idea is use a string as somewhere to keep data of various types. Just as PRINT STRING$ (64,65) will print a line of A's very quickly, so can other quite complex characters be printed remarkably quickly. Although not as quickly as would machine language, which is another use for string packing - to have somewhere to keep a machine language routine without needing to protect memory or other complex manoeuvres. More of this aspect in the machine language episode.

A string can be used as a simple storage location for pieces of data of varying lengths, simply by separating the pieces by some known delimiter (such as CHR$(128)), then using the MID$ function to retrieve the pieces between the delimiters.

Incidentally, we have no idea as to why it is called string packing - there's nothing mysterious at all involved.

INKEY $

This is a very powerful function for getting input without having to press ENTER. It is a routine which scans the keyboard to detect whether a key has been pressed. Since it is so fast, it is necessary to create a loop to use INKEY$ effectively. E.g:

```
100 A$ = INKEY$: IFA$ = ""THEN100
```

which will wait until a key is pressed (any key) before going on to the following program code. As one might expect, the greatest benefit is in checking input, and rejecting it unless it is within the expected range. This will be covered later.

However, one aspect which can cause confusion, whatever the reason for using INKEY$, is that the function itself does not retain the value of the key that was pressed. To retain the value a variable must be assigned, and re-assigned within the loop every time. Without a variable, the program will still wait for a key-press, but you will not be able to use it for an input. The INKEY$ is cleared each time it is accessed, so with a little thought it will become clearer. It is like a flag which is set/reset on passing. We must admit there is room for confusion, so we urge experimentation, as in all things.

The input from INKEY$ is not usually displayed, but this is just a simple matter of using PRINT A$. However, the by-product of this is the use of INKEY$ for a password input, where you do not want observers to observe the secrets. Similar to NOECHO in other versions of BASIC. INKEY$ can also be used to create a timed input. This facility is also available in disk-BASIC, as a real time function of the clock. However, in Level 2 it can be easily created by trial-and-error to give the desired time period. Such a routine would be as a subroutine:

```
100 FOR I = 1 to 1000
200 I$ = INKEY$
300 IF I$ = ""THENNEXT:PRINT"TIME UP":STOP
400 I = 1000 : NEXT
```

Rest of program would follow. The closing of the FOR .... NEXT loop in line 400 is of doubtful and debatable significance, and has been the subject of 4 articles that we have read - more information in back copies of Adelaide Users' News. However, it does no harm, even if the times when it is needed are few. So we follow our earlier-expressed maxim, "when in doubt, do".

Just because the INKEY$ function returns only a one character input, doesn't mean its use is at all limited to this. The string concatenation function can be used:

```
100 I1$ = INKEY$: IF I1$  = ""THEN100ELSEPRINTI1$;
200 I2$ = INKEY$: IF I2$  = ""THEN200ELSEPRINTI2$;
300 I3$ = INKEY$: IF I3$  = ""THEN300ELSEPRINTI3$;
400 I$  = I1$ + I2$ + I3$
```

Of course, between these lines could be a timing loop, check for input, etc., and although this routine would, by its nature, give the right length input, it would be aborted by (accidental) press of the BREAK key. For this reason we have written a routine to provide the disk function of LINE INPUT, which we will present later in its proper and expected place.

If, like us, you have children who just like to practice their typing on the screen, you can create a mini-word-processor with shift arrow capabilities by: 10 A$ = INKEY$: PRINT A$;:GOTO10

CONCLUSION

As a final word, we must implore that input checking is of the most vital significance; while you'll have to wait at least 2 issues for the relevant instalment, in the meantime, we suggest you use string variables for all input, check it by FOR.....NEXT and MID$, and convert it if necessary to a numeric variable by VAL. We would have liked to say more on the subject here, as we've found it's the single major factor for programs to bomb out, but we have already used more space than your Esteemed Editor has allocated us.

Next episode will be on arrays. Before you have time to tremble in terror, let us point out that your telephone book is an array. So is a map, a crossword puzzle; in fact, any groups of information. Even multi-storey buildings.

- 0000000000 -

***** PEEKing (UK) by our U.K. Correspondent Tony Fraser *****

In this first of a number of articles I am going to introduce myself as the UK Correspondent. Although an engineer and physicist by profession, I have studied computing at post-graduate level and in my work I do some programming for mainframe machines. My main interest in computing these days is, however, my personal home computer which is used as a hobby machine, running programmes for geneology (another of my hobbies), post-graduate research and the usual crop of computer games. My children are usually the games players, but Dad enjoys them too. I also have a special interest in the programming of graphics displays which I enjoy as a challenge. My machine is a "Video Genie" (recognised in Australia as a "System 80"). It is modified to include a cassette level meter and the '→' and '←' keys, and I have fitted an external connection to #1 cassette output port for music (noise) output. My output device is a domestic TV, but the next piece of equipment on my list is a printer.

I see my place in this magazine as providing any assistance or service to both UK and other readers which can be more conveniently provided by a UK resident. This will be done by writing regularly on the '80 scene as viewed from the UK, and by providing a contact point in the UK for readers anywhere in the world who may need UK-based contacts. If I can be of any help, simply write to me at the address below. One major task I have agreed to take on is that of providing a postbox for programmes by European readers submitted for publication in MICRO-80. These should now be sent to me for preliminary vetting and copying onto a single cassette to be posted to the editorial staff in Australia for selection for publication. Your original cassettes will then be returned. For those who have not purchased back copies (why not?) and thus may not be entirely clear as to the form of submitting programs, I will give brief details. Programs should be submitted on cassette only (listings are not necessary and listings only are not acceptable). Your covering letter should contain all the information necessary for the running of the program and your address. Your name and address should also appear on the program as a REM statement. Ideally, the only information which should be necessary to run the program is the language, i.e. machine code, BASIC, or TINY PASCAL and Level I or II. Level I programs should be written so as to be compatible with both the TRS-80 and the Video Genie units. We have received very few programs from our UK readers to date, perhaps due to postage difficulties, but now you have no excuse, so get them rolling in. Remember, you get not only fame and immortality in print, but you also get paid for any published programs.

Finally, if you think I could be of any help to you, or you see a useful service I could provide to the readership in general, drop me a line and let me know. My address is:

Tony Edwards, 23 Foxfield Close, Northwood, Middx, United Kingdom.

- 0000000000 -

***** INSTALLERS FOR LOWER CASE MOD. KITS *****

MICRO-80 is seeking experienced technicians or hobbyists in each State, to instal lower case mod. kits for our readers. We will publish the names and addresses of installers each month. Individual readers should negotiate directly with the installers in their States. Although we will only publish the names of installers with adequate credentials (on paper), we can take no responsibility for the standard of service or workmanship offered. We ask individual readers to inform us of their experiences, good or bad, with the installers on our list and we will take appropriate action amend the list, as necessary.

VICTORIA:	Keith Pakenham, 20 Nicholas St. Teesborough, 3173. (03) 798 6162.
	Philip Shufflebotham, 1 Perceval St. Sunbury, 3429. (03) 744 3096.
NEW SOUTH WALES:	Allan Mason, 38 Dresden Ave. Castle Hill, 2154. (02) 680 2538.
	Mr. J. Saxon, 2 Mauldon St. Chifley, ACT. 2606.
WESTERN AUSTRALIA:	Andrew Smallridge, 22 Ravenhill Drive, Thornlie, 6108.

- 0000000000 -

********** SOFTWARE SECTION **********

***** ARITHMETIC L1/4K by C. Stobert *****

Whilst not strictly a teaching program, this program will provide useful practice at addition, subtraction and multiplication for 8 - 10 year old children. First, you are given the choice of operation to be performed and then you are presented with 10 problems, one at a time. You must obtain the correct answer before moving on to the next problem. After the set of 10 problems is completed, your score is displayed and the program offers you the opportunity to start again. The problems are randomly generated so that each pass will be different from the last.

```
1 REM C.STOBERT
2 REM 21 SHERWOOD CRS.
3 REM NTH. DANDENONG VIC. 3175
4 REM 03 795 6590
10 C.
20 P.:P."THIS PROGRAMME IS TO ASSIST YOU
30 P."WITH YOUR ARITHMETIC.
```

```
40 P.:P."DO YOU REQUIRE INSTRUCTIONS ?
50 Y=1:N=0:I."ENTER <Y> FOR YES OR <N> FOR NO";O
60 IFO=0C.:G.200
70 C.:P.:P."YOU DO YOUR PROBLEMS IN THE NORMAL WAY.
80 P."WORKING FROM RIGHT TO LEFT,
90 P."ENTERING EACH PART OF THE ANSWER IN TURN.
100 P."FOR EXAMPLE, IN THE FOLLOWING PROBLEM :-
110 P.T.(20),"1 2 3
120 P.T.(22),"9 7 +
130 P.T.(20),"-----
140 P.T.(21),"?  0
150 P.T.(20),"=====
160 P."YOU WOULD ENTER 2 IN THE TENS COLUMN
170 P."THEN CARRY 1 TO THE HUNDREDS COLUMN.
180 GOS.960:C.:P.:P."   G O O D   L U C K   ! !":P.:P.
200 I."WHO IS TAKING THE TEST ";A$
210 P.:P."WHICH DO YOU WANT (1, 2 OR 3) ?
220 P."ADDITION (+)<1>
225 P."SUBTRACTION (-)<2>
230 I."MULTIPLICATION (X)-<3> ";Z
240 Z=I.(Z):IF(Z<1)+(Z>3)C.:G.210
250 GOS.970
280 T=0:R=1
300 IFZ=1D=R.(499)
400 IFZ=2D=R.(499)+500
500 IFZ=3D=R.(111)
510 E=I.(D/100):F=I.((D-E*100)/10)
520 G=D-100*E-10*F
530 IF(Z=1)+(Z=2)H=R.(499)
540 IFZ=3H=R.(9)
550 I=I.(H/100):J=I.((H-I*100)/10)
560 K=H-100*I-10*J
570 C.:P.AT76,"NOW ";A$
580 P.AT146,:GOS.800
590 A=0:B=0:C=0
600 IFE>0P.AT285,E
610 IF(F>0)+(E>0)P.AT287,F
620 P.AT289,G:IFI>0P.AT349,I
630 IF(J>0)+(I>0)P.AT351,J
640 P.AT353,K;" ";B$
650 F.Y=414TO418:P.ATY,"-":N.Y
660 F.Y=542TO546:P.ATY,"=":N.Y
670 P.AT481,:I.A:GOS.2020
680 P.AT479,:I.B:GOS.2010:GOS.900:IFL<100G.700
690 P.AT477,:I.C:GOS.2000
700 T=T+1:GOS.930:IFL=MP.AT720,:GOS.850:X=1:G.720
710 P.AT720,"SORRY ";A$;" TRY AGAIN":X=0
720 GOS.960:IFX=1R=R+1:IFR>10G.1000
730 IFX=1C.:G.300
740 G.570
800 Y=R.(3):ONYG.810,820,830
810 P."SEE HOW YOU DO":RET.
820 P."HAVE A TRY AT THIS":RET.
830 P."NEXT PROBLEM":RET.
850 Y=R.(3):ONYG.860,870,880
860 P."WELL DONE ";A$:RET.
870 P."KEEP UP THE GOOD WORK ";A$:RET.
880 P."GOOD EFFORT ";A$:RET.
900 IFZ=1L=D+H:RET.
910 IFZ=2L=D-H:RET.
920 IFZ=3L=D*H:RET.
930 M=A+10*B+100*C:RET.
960 P.AT896,:I."PRESS <ENTER> TO CONTINUE ";B$
970 IF Z=1B$="+":RET.
980 IFZ=2B$="-":RET.
990 IFZ=3B$="X":RET.
995 RET.
1000 C.:P.:P."THAT IS 10 PROBLEMS ";A$
1010 P.:P."YOU HAD";T;"TRIES
1020 P."WHICH GAVE YOU A MARK OF";
1030 P.(1-(T-10)/10)*100;"%
1040 P.:P."IF YOU WOULD LIKE TO TRY AGAIN ENTER <1>
1050 I."IF NOT ENTER <2> ";O
1060 C.:IFO=1G.210
1070 P."NOW ";:G.200
2000 P.AT477,C
2010 P.AT479,B
2020 P.AT481,A
2030 RET.
```

***** SORTING L1/4K by Hubb Sanders *****

This is the first program we have published from a reader in Holland. Thank you very much, Hubb. The program demonstrates the action of a "bubble-sort" routine. The objective is to arrange a row of numbers in order with the lowest to the left and the highest to the right. The bubble sort routine achieves this by comparing successive pairs of numbers, starting from the left. If the first number is lower than its neighbour, then their relative positions remain undisturbed and the sort routine passes on. However, if the first number is higher than its neighbour, then the positions of the two numbers are reversed before the sort routine passes on. A pass is completed when each pair has been sorted once. Consider the reconstruction of a screen display for four numbers to be sorted, shown below.

```
ENTER 4 3 2 1
```

Program now displays, one line at a time:

```
PASS 1  SORT 1  3 4 2 1
PASS 1  SORT 2  3 2 4 1
PASS 1  SORT 3  3 2 1 4
PASS 2  SORT 1  2 3 1 4
PASS 2  SORT 2  2 1 3 4
PASS 3  SORT 1  1 2 3 4
        DONE
ONCE MORE (1 = YES  2 = NO)?
```

In PASS 1 SORT 1, the first two digits are interchanged because 4 (the left-most) is higher than 3. In PASS 1 SORT 2 (the second pair) the 4 and 2 are interchanged, and so on.

The name bubble-sort is derived from the way the numbers move. If you imagine that the sorting is carried out on a vertical column of numbers with each pass starting at the top, then the low numbers will move gradually up the column, rather like air bubbles in a fish tank.

The program commences by requesting the total number of inputs. From 8 to 12 inputs are most useful for demonstrating the sort routine - Lines 125 and 546 introduce deliberate delays to enable you to follow the progress of the program. If you would like to see how fast the program can function, delete both these lines.

```
5 C.:P.AT460,"HUBB SANDERS
6 P.AT524,"LUTTERSWEG 36
7 P.AT588,"7782 TA DE KRIM
8 P.AT652,"NETHERLANDS
9 F.B=1TO1000:N.B
10 C.
15 P.AT409,"SORT DEMONSTRATION
16 F.Z=1TO1500:N.Z
20 C.:P."THIS PROGRAM ALLOWS YOU TO INPUT A CERTIAN AMOUNT
21 P.:P."OF NUMBERS AFTER WHICH THE COMPUTER WILL START SORTING.
22 P.:P."WHEN EVER A NUMBER IS MOVED THE PASS WILL BE DISPLAYED.
23 P.:P."TO FREEZE THE DISPLAY HOLD DOWN THE <^> KEY
24 P.AT860,"HIT <ENTER> TO START";:I.A$
30 C.
35 P.AT64,"HOW MANY NUMBERS DO YOU WISH TO INPUT";:I.Q
36 IFQ<2P."IT'S NOT EASY TO SORT THAT MANY":G.35
40 C.:F.K=1TOQ
45 P.AT450,"INPUT NUMBER";K;:I.A(K):C.
50 F.Z=1TOK-1:IFA(Z)<>A(K)N.Z
55 IFZ=KG.70
60 P."YOU'VE ALREADY ENTERED";A(K):G.45
70 N.K
80 F.K=1TOQ-1:F.L=1TOQ-1
90 IFA(L)<A(L+1)G.120
100 P=A(L):A(L)=A(L+1):A(L+1)=P
110 GOS.500
120 N.L
125 F.Z=1TO500:N.Z
130 N.K
135 P.:P.T.(30),"DONE
136 P.T.(20),"ONCE MORE (1=YES 2=NO)";:I.R
137 IFR=1G.30
138 END
500 P."PASS";K;" SORT";L;"    ";
510 F.X=1TOQ
520 P.T.(5),A(X);
530 N.X
540 P.
545 X=X+Q
546 F.Z=1T.250:N.Z
550 RET.
```

***** NORMAL DISTRIBUTION LII/16K (c) T.R. Jones *****

LOADING INSTRUCTIONS.

To load the program from cassette, answer MEMORY SIZE? (READY?) by pressing ENTER/NEWLINE. Then type CLOAD"N" and press ENTER/NEWLINE. When the program has loaded type RUN and press ENTER/NEWLINE.

INTRODUCTION.

This month sees the start of a series of scientific programs written by an author working in a scientific field. This first one is one of the most basic, yet one of the most important and widely used of statistical tools. Many statistical procedures begin with the assumption that data under study approximates this distribution. The normal distribution is such that if one were to plot frequency per class against a range of classes for a given set of observations, e.g. (butterfat production per cow for a dairy herd), then the resulting curve would be approximately bell shaped, having a high point near the middle representing the most common class, sloping off rather symmetrically on either side to the rare exceptional classes on either end. From such a distribution, a series of useful statistical parameters arise. This program will compute these parameters, known as the mean, variance, standard deviation, standard error and the range. The program will accept up to 100 data values for analysis. Following entry of data, values are displayed for verification and an option to edit entries is given. Having entered all the data correctly, the parameters described are computed and displayed on the screen. Here is a brief definition of the parameters.

1. MEAN - mathematical or arithmetic average of the data set.

2. VARIANCE/STANDARD DEVIATION - two ways by which the variability of data about the mean is described.

3. STANDARD ERROR - an estimate of the variability of the mean of the data set from the mean of the population from which the data was drawn.

4. RANGE - is simply the lowest and highest values between which all the data occur.

Simply respond to the program prompts for each type of input.

```
10   'ROUTINE TO CALCULATE STASTICS IN RELATION TO THE NORMAL DISTRIBUTION.

20     '(C) COPYRIGHT 1980, TERRY JONES, 43 HASTIE STREET TATURA, 3616.

60 CLS:DEFINTI-N:DIMX(100),F(10)
70 PRINTTAB(18)"NORMAL DISTRIBUTION ANALYSIS":
   PRINTTAB(18)STRING$(28,45)
80 PRINT:PRINT:PRINTTAB(26)"ENTER DATA":PRINT:PRINTTAB(13)
  "NUMBER OF DATA POINTS - MAXIMUM IS 100"
90 PRINTTAB(28)"*";:INPUTND:IFND<2ORND>100
   PRINTCHR$(27);CHR$(30);CHR$(29);CHR$(27):GOTO90
100 PRINT:FORI=1TOND:PRINTTAB(27)"X(";I;") =";:INPUTX(I):NEXTI
110 CLS:K=0:PRINTTAB(25)"DATA ENTERED":FORI=1TOND:K=K+1:
   PRINTTAB(26)"X(";I;") =";X(I):
   IFK<10NEXTIELSEPRINT@977,"TYPE <ENTER> TO CONTINUE ...";:
   INPUTI$:K=0:CLS:NEXTI
120 PRINT@983,"EDIT DATA (Y/N)";
130 I$=INKEY$:IFI$=""THEN130ELSEIFI$="Y"THEN140ELSEIFI$="N"
   THEN150ELSE130
140 CLS:INPUT"DATA POINT TO BE EDITED";I:
   IFI>NDTHEN140ELSEPRINT"X(";I;") =";X(I):
   PRINTTAB(32)CHR$(27);"NEW X(";I;") =";:INPUTX(I):
   GOTO110
150 CLS:PRINT@473,"COMPUTING ...."
160 X1=0:X2=0:FORI=1TOND:X1=X1+X(I):X2=X2+(X(I)[2):NEXTI:
   X3=X1/ND:X4=X2-(X1[2/ND):X5=X4/(ND-1):X6=SQR(X5):X7=X5/ND
170 R1=X(1):R2=X(1):FORI=1TOND:IFX(I)<R1THENR1=X(I):NEXTI
   ELSEIFX(I)>R2THENR2=X(I):NEXTIELSENEXTI
180 CLS:PRINT:PRINT"MEAN =";X3:PRINT:
   PRINT"VARIANCE =";X5:PRINT:
   PRINT"STANDARD DEVIATION =";X6:PRINT:
   PRINT"STANDARD ERROR =";X7:PRINT:
   PRINT"RANGE =";R1;"TO";R2:PRINT
190 END
```

***** DISASSEMBLER LII/16K (c) Gregg Nott *****

LOADING INSTRUCTIONS.

If this program is being typed in from the magazine, type it in * EXACTLY * as listed and CSAVE it to tape/wafer BEFORE you run it. When loading the program from tape answer MEMORY SIZE? (READY?) with 27740 and press ENTER/NEWLINE then type CLOAD"D" and press ENTER/NEWLINE. When the program has loaded type RUN and press ENTER/NEWLINE.

NOTE: When using this program if the address supplied by you is not the first byte of a valid Z80 instruction or is an area where data is stored, an error message will be given.

INTRODUCTION.

This disassembler outputs standard Z-80 mnemonics from any block of machine code in memory. The program can disassemble the entire Z-80 instruction set, i.e. provide the appropriate mnemonic representation for all 700 odd instructions. The program leaves approximately 6K bytes of high memory in which the desired machine language program to be disassembled, can reside. Naturally ROM and dedicated addresses can also be disassembled. The program commences by asking for a starting address in decimal. Fourteen lines, representing one instruction per line, are then displayed on the video screen. The current address in hexidecimal is printed on the far left followed by the actual contents appropriate for the instruction. A feature of the program is that data and addresses associated with instructions are displayed with the mnemonics (including relative jump addresses). In addition ASCII characters (letters) are printed on the far right of the display, to aid in the identification of messages (DEFM) etc. When the ENTER/NEWLINE key is pressed the screen is cleared and fourteen more lines are printed. The program is terminated by pressing the BREAK key.

REM lines have been placed in the program in order to identify subroutines and to facilitate an understanding of its operation. REM lines should be removed if more memory is required. (Answer MEMORY SIZE? with 27000 in this case). One word of caution, lines up to and including line 50 should be typed in exactly as shown. LIne 20 and 50 contain GOSUB statements which have their argument (line number) changed as every instruction is disassembled. (ED: before you all write in, that's why we didn't renumber this program as it POKEs that are of the BASIC program and when the program was renumbered, (yes we tried), it POKEd itself into the middle of next week!) A routine calculates the correct line number and POKEs the values into the program storage area immediately after the GOSUB token. This explains the GOSUB XXXX statements found in the listing before it has been RUN.

You may set the MEMORY SIZE to either of the following values to protect your high memory m/l program:

With REM lines in place (as listed) set to - 27740

With REM lines removed set to - 27000

If you really want to know how this program works, then read on, otherwise skip the rest and start using it. The principal technique used in the program is a routine that modifies a section of the program as it resides in the program storage area, i.e. 42E9H and upwards. The program manipulates only two areas of itself, in lines 20 and 50, after the BASIC GOSUB tokens. In standard BASIC the GOSUB statement must be followed by a constant, i.e. a line number. In effect, the self modifying routine allows the argument of GOSUB to be a variable under program control. In lines 20 and 50 GOSUB becomes a conditional statement with multiple branchings, with a structure somewhat similar to the ON X GOSUB statement.

DISASSEMBLER commences by requesting a starting address, from which disassembly will commence. Using the given value as a pointer, the program starts PEEKing into memory (note: this value must be the FIRST byte of a valid instruction or disassembly will be thrown out of synchronization). The PEEKed value is processed by routines (lines 2110-2150) that modify the program storage area in line 20. There are 255 possible branches from line 20 (lines 1000-1510) corresponding to all possible first byte values. Variable L, the length of instruction indicator, is set by every line that is branched to, to correspond with the length of the Z-80 instruction. Upon branching, the appropriate Z-80 mnemonics are PRINTed and calls are made to appropriate subroutines to PRINT data, relative displacements and addresses associated with the instruction. Program flow then returns to the GOSUB statement where the memory pointer is incremented (variable L) and a call is made to a subroutine that PRINTs the machine codes representing the instruction (lines 10000-10010).

In addition to the various subroutines mentioned, there are 4 conditional exits from the primary program loop described. These are necessary to accommodate the large number of Z-80 instructions that commence with a CB,DD,ED and FD byte value. DD,ED and FD instructions all use a secondary self modifying loop that branches from line 50, while CB instructions are handled by a number of simple algorithms (lines 10020-10220). These instructions use the PEEKed value of the SECOND byte in the instruction to process the correct branching line number from the GOSUB in line 50 (certain DD and FD 4 byte indexed instructions PEEK at the LAST byte of the instruction in order to differentiate between instructions).

SUBROUTINES, SIGNIFICANT VARIABLES AND PROGRAM BLOCKS.

PROGRAM LINES	SUBROUTINES
2119 - 2130	CALCULATES BRANCHING LINE NUMBER.
2150	POKES LINE NUMBER INTO PROGRAM.
3000 - 3010	PRINTS CURRENT ADDRESS IN HEX.
4000 - 4700	4 SUBROUTINES FOR PRINTING DATA OR ADDRESS ASSOCIATED WITH INSTRUCTIONS.
5000 - 5120	DECIMAL TO HEXADECIMAL CONVERSION.
5200 - 5230	CALCULATING & PRINTING DISPLACEMENT VALUES.
10000 - 10010	PRINT MEMORY ASSOCIATED WITH INSTRUCTION.
10012 - 11100	PEEK AT 2ND BYTE OF DD/ED/FD INSTRUCTIONS.
12000 -	ERROR TRAPPING ROUTINE FOR DETECTING UL ERRORS RESULTING FROM CB/DD/ED/FD DATA VALUES.

MAJOR VARIABLES.

Z	PEEKed memory address
A	Decimal value of variable Z (memory content)
L	Length of instruction indicator
K	Video display line counter
C$	Output from dec-hex conversion
E	Input to Dec-hex conversion
M	Pointer to program storage area where line number will be POKEd (Y1/Y2/Y3/Y4)
Y$	Variable A converted to GOSUB line number.

MAJOR PROGRAM BLOCKS/POINTS

1000 - 1510	PRIMARY LOOP PRINT LINES
2005	STARTING POINT OF PRIMARY LOOP
1442 -	SETUP DD INSTRUCTION SECONDARY LOOP
1506 -	SETUP FD INSTRUCTION SECONDARY LOOP
8000 - 8010	SETUP ED INSTRUCTION SECONDARY LOOP
6064 - 6187	ED INSTRUCTION PRINT LINES
7009 - 7249	DD AND FD INSTRUCTION PRINT LINES
10020 - 10220	CB INSTRUCTION PROCESSING

```
 10 CLEAR100:CLS
 15 DEFINTA-Z:L=0:K=0:GOSUB2000
 18 GOTO2005
 20 GOSUBXXXX
 25 IFA>62ANDA<194THENL=1
 30 GOSUB10000:Z=Z+L:GOTO2005
 50 GOSUBXXXX
 60 RETURN
 100 REM ** Z80 DISASSEMBLER   BY GREG NOTT 20/2/80 **
 1000 L=1:PRINT"NOP";:RETURN
 1002 L=3:PRINT"LD BC,";:GOSUB4500:RETURN
 1004 L=1:PRINT"LD (BC),A";:RETURN
 1006 L=1:PRINT"INC BC";:RETURN
 1008 L=1:PRINT"INC B";:RETURN
 1010 L=1:PRINT"DEC B";:RETURN
 1012 L=2:PRINT"LD B,";:GOSUB4000:RETURN
 1014 L=1:PRINT"RLC A";:RETURN
 1016 L=1:PRINT"EX AF,AF'";:RETURN
 1018 L=1:PRINT"ADD HL,BC";:RETURN
 1020 L=1:PRINT"LD A,(BC)";:RETURN
 1022 L=1:PRINT"DEC BC";:RETURN
 1024 L=1:PRINT"INC C";:RETURN
 1026 L=1:PRINT"DEC C";:RETURN
 1028 L=2:PRINT"LD C,";:GOSUB4000:RETURN
 1030 L=1:PRINT"RRC A";:RETURN
 1032 L=2:PRINT"DJ NZ ";:GOTO5200
 1034 L=3:PRINT"LD DE,";:GOSUB4500:RETURN
 1036 L=1:PRINT"LD (DE),A";:RETURN
 1038 L=1:PRINT"INC DE";:RETURN
 1040 L=1:PRINT"INC D";:RETURN
 1042 L=1:PRINT"DEC D";:RETURN
 1044 L=2:PRINT"LD D,";:GOSUB4000:RETURN
 1046 L=1:PRINT"RL A";:RETURN
 1048 L=2:PRINT"JR ";:GOTO5200
 1050 L=1:PRINT"ADD HL,DE";:RETURN
 1052 L=1:PRINT"LD A,(DE)";:RETURN
```

```
1054 L=1:PRINT"DEC DE";:RETURN
1056 L=1:PRINT"INC E";:RETURN
1058 L=1:PRINT"DEC E";:RETURN
1060 L=2:PRINT"LD E,";:GOSUB4000:RETURN
1062 L=1:PRINT"RR A";:RETURN
1064 L=2:PRINT"JR NZ ";:GOTO5200
1066 L=3:PRINT"LD HL,";:GOSUB4500:RETURN
1068 L=3:PRINT"LD (";:GOSUB4500:PRINT"),HL";:RETURN
1070 L=1:PRINT"INC HL";:RETURN
1072 L=1:PRINT"INC H";:RETURN
1074 L=1:PRINT"DEC H";:RETURN
1076 L=2:PRINT"LD H,";:GOSUB4000:RETURN
1078 L=1:PRINT"DAA";:RETURN
1080 L=2:PRINT"JR Z ";:GOTO5200
1082 L=1:PRINT"ADD HL,HL";:RETURN
1084 L=3:PRINT"LD HL,(";:GOSUB4500:PRINT")";:RETURN
1086 L=1:PRINT"DEC HL";:RETURN
1088 L=1:PRINT"INC L";:RETURN
1090 L=1:PRINT"DEC L";:RETURN
1092 L=2:PRINT"LD L,";:GOSUB4000:RETURN
1094 L=1:PRINT"CPL";:RETURN
1096 L=2:PRINT"JR NC ";:GOTO5200
1098 L=3:PRINT"LD SP,";:GOSUB4500:RETURN
1100 L=3:PRINT"LD (";:GOSUB4500:PRINT"),A";:RETURN
1102 L=1:PRINT"INC SP";:RETURN
1104 L=1:PRINT"INC (HL)";:RETURN
1106 L=1:PRINT"DEC (HL)";:RETURN
1108 L=2:PRINT"LD (HL),";:GOSUB4000:RETURN
1110 L=1:PRINT"SCF";:RETURN
1112 L=2:PRINT"JR C ";:GOTO5200
1114 L=1:PRINT"ADD HL,SP";:RETURN
1116 L=3:PRINT"LD A,(";:GOSUB4500:PRINT")";:RETURN
1118 L=1:PRINT"DEC SP";:RETURN
1120 L=1:PRINT"INC A";:RETURN
1122 L=1:PRINT"DEC A";:RETURN
1124 L=2:PRINT"LD A,";:GOSUB4000:RETURN
1126 PRINT"CCF";:RETURN
1128 PRINT"LD B,B";:RETURN
1130 PRINT"LD B,C";:RETURN
1132 PRINT"LD B,D";:RETURN
1134 PRINT"LD B,E";:RETURN
1136 PRINT"LD B,H";:RETURN
1138 PRINT"LD B,L";:RETURN
1140 PRINT"LD B,(HL)";:RETURN
1142 PRINT"LD B,A";:RETURN
1144 PRINT"LD C,B";:RETURN
1146 PRINT"LD C,C";:RETURN
1148 PRINT"LD C,D";:RETURN
1150 PRINT"LD C,E";:RETURN
1152 PRINT"LD C,H";:RETURN
1154 PRINT"LD C,L";:RETURN
1156 PRINT"LD C,(HL)";:RETURN
1158 PRINT"LD C,A";:RETURN
1160 PRINT"LD D,B";:RETURN
1162 PRINT"LD D,C";:RETURN
1164 PRINT"LD D,D";:RETURN
1166 PRINT"LD D,E";:RETURN
1168 PRINT"LD D,H";:RETURN
1170 PRINT"LD D,L";:RETURN
1172 PRINT"LD D,(HL)";:RETURN
1174 PRINT"LD D,A";:RETURN
1176 PRINT"LD E,B";:RETURN
1178 PRINT"LD E,C";:RETURN
1180 PRINT"LD E,D";:RETURN
1182 PRINT"LD E,E";:RETURN
1184 PRINT"LD E,H";:RETURN
1186 PRINT"LD E,L";:RETURN
1188 PRINT"LD E,(HL)";:RETURN
1190 PRINT"LD E,A";:RETURN
1192 PRINT"LD H,B";:RETURN
1194 PRINT"LD H,C";:RETURN
1196 PRINT"LD H,D";:RETURN
1198 PRINT"LD H,E";:RETURN
1200 PRINT"LD H,H";:RETURN
1202 PRINT"LD H,L";:RETURN
1204 PRINT"LD H,(HL)";:RETURN
```

```
1206 PRINT"LD H,A";:RETURN
1208 PRINT"LD L,B";:RETURN
1210 PRINT"LD L,C";:RETURN
1212 PRINT"LD L,D";:RETURN
1214 PRINT"LD L,E";:RETURN
1216 PRINT"LD L,H";:RETURN
1218 PRINT"LD L,L";:RETURN
1220 PRINT"LD L,(HL)";:RETURN
1222 PRINT"LD L,A";:RETURN
1224 PRINT"LD (HL),B";:RETURN
1226 PRINT"LD (HL),C";:RETURN
1228 PRINT"LD (HL),D";:RETURN
1230 PRINT"LD (HL),E";:RETURN
1232 PRINT"LD (HL),H";:RETURN
1234 PRINT"LD (HL),L";:RETURN
1236 PRINT"HALT";:RETURN
1238 PRINT"LD (HL),A";:RETURN
1240 PRINT"LD A,B";:RETURN
1242 PRINT"LD A,C";:RETURN
1244 PRINT"LD A,D";:RETURN
1246 PRINT"LD A,E";:RETURN
1248 PRINT"LD A,H";:RETURN
1250 PRINT"LD A,L";:RETURN
1252 PRINT"LD A,(HL)";:RETURN
1254 PRINT"LD A,A";:RETURN
1256 PRINT"ADD A,B";:RETURN
1258 PRINT"ADD A,C";:RETURN
1260 PRINT"ADD A,D";:RETURN
1262 PRINT"ADD A,E";:RETURN
1264 PRINT"ADD A,H";:RETURN
1266 PRINT"ADD A,L";:RETURN
1268 PRINT"ADD A,(HL)";:RETURN
1270 PRINT"ADD A,A";:RETURN
1272 PRINT"ADC A,B";:RETURN
1274 PRINT"ADC A,C";:RETURN
1276 PRINT"ADC A,D";:RETURN
1278 PRINT"ADC A,E";:RETURN
1280 PRINT"ADC A,H";:RETURN
1282 PRINT"ADC A,L";:RETURN
1284 PRINT"ADC A,(HL)";:RETURN
1286 PRINT"ADC A,A";:RETURN
1288 PRINT"SUB B";:RETURN
1290 PRINT"SUB C";:RETURN
1292 PRINT"SUB D";:RETURN
1294 PRINT"SUB E";:RETURN
1296 PRINT"SUB H";:RETURN
1298 PRINT"SUB L";:RETURN
1300 PRINT"SUB (HL)";:RETURN
1302 PRINT"SUB A";:RETURN
1304 PRINT"SBC A,B";:RETURN
1306 PRINT"SBC A,C";:RETURN
1308 PRINT"SBC A,D";:RETURN
1310 PRINT"SBC A,E";:RETURN
1312 PRINT"SBC A,H";:RETURN
1314 PRINT"SBC A,L";:RETURN
1316 PRINT"SBC A,(HL)";:RETURN
1318 PRINT"SBC A,A";:RETURN
1320 PRINT"AND B";:RETURN
1322 PRINT"AND C";:RETURN
1324 PRINT"AND D";:RETURN
1326 PRINT"AND E";:RETURN
1328 PRINT"AND H";:RETURN
1330 PRINT"AND L";:RETURN
1332 PRINT"AND (HL)";:RETURN
1334 PRINT"AND A";:RETURN
1336 PRINT"XOR B";:RETURN
1338 PRINT"XOR C";:RETURN
1340 PRINT"XOR D";:RETURN
1342 PRINT"XOR E";:RETURN
1344 PRINT"XOR H";:RETURN
1346 PRINT"XOR L";:RETURN
1348 PRINT"XOR (HL)";:RETURN
1350 PRINT"XOR A";:RETURN
1352 PRINT"OR B";:RETURN
1354 PRINT"OR C";:RETURN
1356 PRINT"OR D";:RETURN
```

```
1358 PRINT"OR E";:RETURN
1360 PRINT"OR H";:RETURN
1362 PRINT"OR L";:RETURN
1364 PRINT"OR (HL)";:RETURN
1366 PRINT"OR A";:RETURN
1368 PRINT"CP B";:RETURN
1370 PRINT"CP C";:RETURN
1372 PRINT"CP D";:RETURN
1374 PRINT"CP E";:RETURN
1376 PRINT"CP H";:RETURN
1378 PRINT"CP L";:RETURN
1380 PRINT"CP (HL)";:RETURN
1382 PRINT"CP A";:RETURN
1384 PRINT"RET NZ";:RETURN
1386 PRINT"POP BC";:RETURN
1388 L=3:PRINT"JP NZ,";:GOSUB4500:RETURN
1390 L=3:PRINT"JP ";:GOSUB4500:RETURN
1392 L=3:PRINT"CALL NZ,";:GOSUB4500:RETURN
1394 L=1:PRINT"PUSH BC";:RETURN
1396 L=2:PRINT"ADD A,";:GOSUB4000:RETURN
1398 L=1:PRINT"RST 0";:RETURN
1400 L=1:PRINT"RET Z";:RETURN
1402 L=1:PRINT"RET";:RETURN
1404 L=3:PRINT"JP Z,";:GOSUB4500:RETURN
1406 GOTO10020
1408 L=3:PRINT"CALL Z,";:GOSUB4500:RETURN
1410 L=3:PRINT"CALL ";:GOSUB4500:RETURN
1412 L=2:PRINT"ADC A,";:GOSUB4000:RETURN
1414 L=1:PRINT"RST 8";:RETURN
1416 L=1:PRINT"RET NC";:RETURN
1418 L=1:PRINT"POP DE";:RETURN
1420 L=3:PRINT"JP NC,";:GOSUB4500:RETURN
1422 L=2:PRINT"OUT ";:GOSUB4000:PRINT",A";:RETURN
1424 L=3:PRINT"CALL NC,";:GOSUB4500:RETURN
1426 L=1:PRINT"PUSH DE";:RETURN
1428 L=2:PRINT"SUB ";:GOSUB4000:RETURN
1430 L=1:PRINT"RST 10";:RETURN
1432 L=1:PRINT"RET C";:RETURN
1434 L=1:PRINT"EXX";:RETURN
1436 L=3:PRINT"JP C,";:GOSUB4500:RETURN
1438 L=2:PRINT"IN A,";:GOSUB4000:RETURN
1440 L=3:PRINT"CALL C,";:GOSUB4500:RETURN
1442 G$="IX":P$="(IX+IND)":GOSUB11000:Y$=RIGHT$(STR$(F+7000),4)
1443 GOSUB2110:M=17231:GOSUB2150:GOTO50
1444 L=2:PRINT"SBC A,";:GOSUB4000:RETURN
1446 L=1:PRINT"RST 18";:RETURN
1448 L=1:PRINT"RET PO";:RETURN
1450 L=1:PRINT"POP HL";:RETURN
1452 L=3:PRINT"JP PO,";:GOSUB4500:RETURN
1454 L=1:PRINT"EX (SP),HL";:RETURN
1456 L=3:PRINT"CALL PO,";:GOSUB4500:RETURN
1458 L=1:PRINT"PUSH HL";:RETURN
1460 L=2:PRINT"AND ";:GOSUB4000:RETURN
1462 L=1:PRINT"RST 20";:RETURN
1464 L=1:PRINT"RET PE";:RETURN
1466 L=1:PRINT"JP (HL)";:RETURN
1468 L=3:PRINT"JP PE,";:GOSUB4500:RETURN
1470 L=1:PRINT"EX DE,HL";:RETURN
1472 L=3:PRINT"CALL PE,";:GOSUB4500:RETURN
1474 GOTO8000
1476 L=2:PRINT"XOR ";:GOSUB4000:RETURN
1478 L=1:PRINT"RST 28";:RETURN
1480 L=1:PRINT"RET P";:RETURN
1482 L=1:PRINT"POP AF";:RETURN
1484 L=3:PRINT"JP P,";:GOSUB4500:RETURN
1486 L=1:PRINT"DI";:RETURN
1488 L=3:PRINT"CALL P,";:GOSUB4500:RETURN
1490 L=1:PRINT"PUSH AF";:RETURN
1492 L=2:PRINT"OR ";:GOSUB4000:RETURN
1494 L=1:PRINT"RST 30";:RETURN
1496 L=1:PRINT"RET M";:RETURN
1498 L=1:PRINT"LD SP,HL";:RETURN
1500 L=3:PRINT"JP M,";:GOSUB4500:RETURN
1502 L=1:PRINT"EI";:RETURN
1504 L=3:PRINT"CALL M,";:GOSUB4500:RETURN
1506 G$="IY":P$="(IY+IND)":GOSUB11000:Y$=RIGHT$(STR$(F+7000),4)
```

```
1507 GOSUB2110:M=17231:GOSUB2150:GOTO50
1508 L=2:PRINT"CP ";:GOSUB4000:RETURN
1510 L=1:PRINT"RST 38";:RETURN
2000 INPUT"ENTER STARTING ADDRESS";Z:CLS:RETURN
2002 ONERRORGOTO12000:RETURN
2005 IFK=14THENINPUT"PRESS ENTER TO CONTINUE";A$:CLS:K=0
2010 A=PEEK(Z):K=K+1:Y$=RIGHT$(STR$(A*2+1000),4)
2020 GOSUB2110:M=17178:GOSUB2150:V=Z:GOSUB3000:GOTO20
2099 REM ** DETERMINE WHAT TO POKE  ENTRANCE Y$ **
2110 Y1$=RIGHT$(Y$,1):Y2$=MID$(Y$,3,1)
2120 Y3$=MID$(Y$,2,1):Y4$=LEFT$(Y$,1)
2130 Y1=ASC(Y1$):Y2=ASC(Y2$):Y3=ASC(Y3$):Y4=ASC(Y4$):RETURN
2149 REM ** POKE IT  >M< =START **
2150 POKEM,Y4:POKEM+1,Y3:POKEM+2,Y2:POKEM+3,Y1:RETURN
2999 REM ** PRINT CURRENT ADDRESS **
3000 A1=V/256:A2=V-A1*256
3010 E=A1:GOSUB5000:PRINTC$;:E=A2:GOSUB5000:PRINTC$,,:RETURN
3999 REM ** PEEK & PRINT A SINGLE BYTE **
4000 E=PEEK(Z+1):GOSUB5000:PRINTC$;:RETURN
4499 REM ** PEEK & PRINT A DOUBLE BYTE **
4500 E=PEEK(Z+2):GOSUB5000:PRINTC$;:GOSUB4000:RETURN
4599 REM ** PEEK & PRINT DOUBLE BYTE FROM 4 BYTE INSTR. **
4600 E=PEEK(Z+3):GOSUB5000:PRINTC$;:E=PEEK(Z+2):GOSUB5000
4610 PRINTC$;:RETURN
4699 REM ** PEEK & PRINT SINGLE BYTE FROM 4 BYTE INSTR. **
4700 E=PEEK(Z+3):GOSUB5000:PRINTC$;:RETURN
4999 REM ** DEC. TO HEX. CONVERSION **
5000 C$="":B=E/16:D=B:IFB>9THEN5050ELSEC$=RIGHT$(STR$(B),1)
5020 X=E-B*16:D=X:IFX>9THEN5050ELSEC$=C$+RIGHT$(STR$(X),1)
5030 RETURN
5050 OND-9GOTO5060,5070,5080,5090,5100,5110
5060 C$=C$+"A":GOTO5120
5070 C$=C$+"B":GOTO5120
5080 C$=C$+"C":GOTO5120
5090 C$=C$+"D":GOTO5120
5100 C$=C$+"E":GOTO5120
5110 C$=C$+"F"
5120 IFLEN(C$)=2THENRETURNELSEGOTO5020
5199 REM ** CALCULATING & PRINTING DISPLACEMENTS (JR INSTR.) *
5200 V=PEEK(Z+1):I=Z:IFV>127THENGOTO5220ELSEV1=I+V+2:GOTO5230
5220 V1=I+(V-256)+2
5230 PRINT" ";:V=V1:GOSUB3000:GOTO30
5999 REM ** PRINT ED INSTRUCTIONS **
6064 PRINT"IN B,(C)";:RETURN
6065 PRINT"OUT (C),B";:RETURN
6066 PRINT"SBC HL,BC";:RETURN
6067 PRINT"LD (";:GOSUB4600:PRINT"),BC";:RETURN
6068 PRINT"NEG";:RETURN
6069 PRINT"RET N";:RETURN
6070 PRINT"IM O";:RETURN
6071 PRINT"LD 1,A";:RETURN
6072 PRINT"IN C,(C)";:RETURN
6073 PRINT"OUT (C),C";:RETURN
6074 PRINT"ADC HL,BC";:RETURN
6075 PRINT"LD BC,(";:GOSUB4600:PRINT")";:RETURN
6077 PRINT"RET I";:RETURN
6080 PRINT"IN D,(C)";:RETURN
6081 PRINT"OUT (C),D";:RETURN
6082 PRINT"SBC HL,DE";:RETURN
6083 PRINT"LD (";:GOSUB4600:PRINT"),DE";:RETURN
6086 PRINT"IM I";:RETURN
6087 PRINT"LD A,I";:RETURN
6088 PRINT"IN E,(C)";:RETURN
6089 PRINT"OUT (C),E";:RETURN
6090 PRINT"ADC HL,DE";:RETURN
6091 PRINT"LD DE,(";:GOSUB4600:PRINT")";:RETURN
6094 PRINT"IM 2";:RETURN
6095 PRINT"LD A,R";:RETURN
6096 PRINT"IN H,(C)";:RETURN
6097 PRINT"OUT (C),H";:RETURN
6098 PRINT"SBC HL,HL";:RETURN
6103 PRINT"RRD";:RETURN
6104 PRINT"IN L,(C)";:RETURN
6105 PRINT"OUT (C),L";:RETURN
6106 PRINT"ADC HL,HL";:RETURN
6111 PRINT"RLD";:RETURN
```

DON'T BE HELD BACK BY AN ANTIQUATED DISK OPERATING SYSTEM

MOVE UP TO

NEWDOS 80 **$149 incl. p&p**

NEWDOS 80 is a completely new DOS for the TRS-80 SYSTEM 80. It is well-documented, bug free and increases the power of your system many times over. It is upward compatible with TRSDOS AND NEWDOS (ie TRSDOS and NEWDOS+ programs will run on NEWDOS 80 but the reverse is not necessarily so).

These are just a few of the many new features offered by NEWDOS 80.

* New BASIC commands that support variable record lengths up to 4095 bytes long.
* Mix or match disk drives. Supports any track count from 18 to 96. Use 35, 40, 77 or 80 track 5¼ inch mini disk drives, 8 inch disk drives OR ANY COMBINATION.
* An optional security boot-up for BASIC or machine code application programs. User never sees "DOS-READY" or "READY" and is unable to "BREAK", clear screen or issue any direct BASIC statements, including "LIST".
* New editing commands that allow program lines to be deleted from one location and moved to another or to allow the duplication of a program line with the deletion of the original.
* Enhanced and improved RENUMBER that allows relocation of subroutines.
* Create powerful chain command files which will control the operation of your system.
* Device handling for routing to display and printer simultaneously.
* MINIDOS — striking the D, F and G keys simultaneously calls up a MINIDOS which allows you to perform many of the DOS commands without disturbing the resident program.
* Includes Superzap 3.0 which enables you to display/print/modify any byte in memory or on disk.
* Also includes the following utilities:
  - Disk Editor/Assembler
  - Disassembler (Z80 machine code)
  - LM offset — allows transfers of any system tape to Disk file — automatically relocated.
  - LEVEL I — Lets you convert your computer back to Level 1.
  - LVIDKSL — Saves and loads Level 1 programs to disk.
  - DIRCHECK — Tests disk directories for errors and lists them.
  - ASPOOL — An automatic spooler which routes a disk file to the printer whilst the computer continues to operate on other programs.
  - LCDVR — a lower case drives which display lower case on the screen if you have fitted a simple lower case modification.

DISK DRIVE USERS ELIMINATE CRC ERRORS AND TRACK LOCKED OUT MESSAGES FIT A PERCOM DATA SEPARATOR

$37.00 plus $1.20 p&p.

When Tandy designed the TRS-80 expansion interface, they did not include a data separator in the disk-controller circuitry, despite the I.C. manufacturer's recommendations to do so. The result is that many disk drive owners suffer a lot of Disk I/O errors. The answer is a data separator. This unit fits inside your expansion interface. It is supplied with full instructions and is a must for the serious disk user.

MPI DISK DRIVES HIGHER PERFORMANCE — LOWER PRICE

MPI is the second largest manufacturer of disk drives in the world. MPI drives use the same form of head control as 8" drives and consequently, they have the fastest track-to-track access time available — 5msec! All MPI drives are capable of single or double-density operation. Double-density operation requires the installation of a PERCOM doubler board in the expansion interface.

As well as single head drives, MPI also makes dual-head drives. A dual-head drive is almost as versatile as two single-head drives but is much cheaper.

Our MPI drives are supplied bare or in a metal cabinet — set up to operate with your TRS-80 or SYSTEM 80. All drives are sold with a 90 day warranty and service is available through MICRO-80 PRODUCTS.

MPI B51 40 Track Single Head Drive. . . . . .only $339
MPI B52 40 Track Double Head Drive. . . . .only $449

Prices are for bare drives and include p&p. Add $10.00 per drive for a cabinet and $60.00 for a power supply to suit two drives. 40 track drives are entirely compatible with 35 track drives. A 40 track DOS such as NEWDOS 80 is necessary to utilise the extra 5 tracks.

OVER 800 KILOBYTES ON ONE DISKETTE! WITH MPI 80 TRACK DRIVES

MPI 80 track drives are now available. The B91 80 track single-head drive stores 204 Kilobytes of formatted data on one side of a 5¼ inch diskette in single-density mode. In double-density mode it stores 408 Kilobytes and loads/saves data twice as quickly.

The B92 80 track dual-head drive stores 204 Kilobytes of formatted data on EACH side of a 5¼ inch diskette in single-density mode. That's 408 Kilobytes per diskette. In double-density mode, the B92 stores a mammoth 408 Kilobytes per side or 816 Kilobytes of formatted data per diskette. With two B92's and a PERCOM double, you could have over 1.6 Megabytes of on line storage for your TRS-80 for less than $1500!!

MPI B91 80 Track Single Head Drive. . . . . .only $499
MPI B92 80 Track Dual Head Drive . . . . . .only $599

Prices are for bare drives and include p&p. Add $10.00 per drive for a cabinet and $60.00 for a power supply to suit two drives. Note: 80 track drives will not read diskettes written on a 35 or 40 track drive. If drives with different track counts are to be operated on the same system, NEWDOS 80 must be used.

CARE FOR YOUR DISK DRIVES? THEN USE 3M's DISK DRIVE HEAD CLEANING DISKETTES

$30.20 incl. p&p.

Disk drives are expensive and so are diskettes. As with any magnetic recording device, a disk drive works better and lasts longer if the head is cleaned regularly. In the past, the problem has been, how do you clean the head without pulling the mechanism apart and running the risk of damaging delicate parts. 3M's have come to our rescue with SCOTCH BRAND, non-abrasive, head cleaning diskettes which thoroughly clean the head in seconds. The cleaning action is less abrasive than an ordinary diskette and no residue is left behind. Each kit contains:

- 2 head cleaning diskettes
- 1 bottle of cleaning fluid
- 1 bottle dispenser cap

PLEASE USE ORDER FORM ON PAGE 36

USE TANDY PERIPHERALS ON YOUR SYSTEM-80 VIA SYSPAND-80 – $119 incl. p&p

The SYSTEM-80 hardware is not compatible with the TRS-80 in two important areas. The printer port is addressed differently and the expansion bus is entirely different. This means that SYSTEM-80 owners are denied the wealth of economical, high performance peripherals which have been developed for the TRS-80. Until now, that is. MICRO-80 has developed the SYSPAND-80 adaptor to overcome this problem.- A completely self-contained unit in a small cabinet which matches the colour scheme of your computer, it connects to the 50-way expansion part on the rear of your SYSTEM 80 and generates the FULL Tandy 40 way bus as well as providing a Centronics parallel printer port. SYSPAND-80 enables you to run an Exatron Stringy Floppy from your SYSTEM 80, or an LNW Research expansion interface or any other desirable peripherals designed to interface to the TRS-80 expansion port. Make your SYSTEM 80 hardware compatible with the TRS-80 via SYSPAND-80.

PROGRAMS BY MICROSOFT

EDITOR ASSEMBLER PLUS (L2/16K) $37.50 + $1.20 p&p

A much improved editor-assembler and debug/monitor for L2/16K TRS-80 or SYSTEM 80. Assembles directly into memory, supports macros and conditional assembly, includes new commands-substitute, move, copy and extend.

LEVEL III BASIC $59.95 plus $1.20 p&p

Loads on top of Level II BASIC and gives advanced graphics, automatic renumbering, single stroke instructions (shift-key entries) keyboard debounce, suitable for L2/16K and up (Not Disk BASIC)

ADVENTURE ON DISK $35.95 plus $1.20 p&p

This is the original ADVENTURE game adapted for the TRS-80. The game fills an entire diskette. Endless variety and challenge as you seek to rise to the level of Grand Master. Until you gain skill, there are whole areas of the cave that you cannot enter. (Requires 32K One Disk)

BASIC COMPILER $208 plus $2.00 p&p

New improved version, the Basic Compiler converts Disk BASIC programs to machine code, automatically. A compiled program runs, on average, 3-10 times faster than the original BASIC program and is much more difficult to pirate.

UPGRADE TO 16K FOR ONLY $30.00!!

MICRO-80's 16K MEMORY EXPANSION KIT HAS BEEN REDUCED IN PRICE EVEN MORE

Larger volume means we buy better and we pass the savings on to you. These are our proven, prime, branded 200 ns (yes, 200 nanosecond) chips. You will pay much more elsewhere for slow, 350 ns. chips. Ours are guaranteed for 12 months. A pair of DIP shunts is also required to upgrade the CPU memory in the TRS-80 – these cost an additional $4.00. All kits come complete with full, step-by-step instructions which include labelled photographs. No soldering is required. You do not have to be an experienced electronic technician to instal them.

DISK DRIVE CABLES SUITABLE FOR ANY DISK DRIVES

DC-2 2 Drive Connector Cable . . . . . $35 incl. p&p
DC-4 4 Drive Connector Cable . . . . . $45 incl. p&p

DOUBLE THE SPEED AND CAPACITY OF YOUR DISK DRIVES PERCOM DOUBLER ONLY $220 plus $2.00 p&p

Installing a Doubler is like buying another set of disk drives, only much cheaper!! The doubler works with most modern disk drives including:- MPI, Micropolis, Pertec, TEAC (as supplied by Tandy). The doubler installs in the TRS-80 expansion interface, the System-80 expansion interface and the LNW Research expansion interface in a few minutes without any soldering, cutting of tracks, etc. It comes complete with its own TRSDOS compatible double density operating system.

DOUBLE-ZAP II – DOUBLE DENSITY PATCH FOR NEWDOS 80 ONLY $53.00 plus $1.00 p&p

If you are using NEWDOS 80, then you also need DOUBLE-ZAP II, on diskette. This program upgrades your NEWDOS 80 to double density with ADR (automatic density recognition.) It retains all the familiar features, including the ability to mix and match track counts on the same cable. In addition, it gives NEWDOS 80 the ability to mix densities on the same cable, automatically. If you place a single density diskette in drive Ø, say and a double density diskette in drive 1, Double-ZapII will recognise this and read/write to drive Ø in single density whilst at the same time it reads/writes to drive 1 in double density!

FLOPPY DOCTOR AND MEMORY DIAGNOSTIC (by MICRO CLINIC) $29.95 plus 50c. p&p

Two machine language programs on a diskette together with manual which thoroughly test your disk drives and memory. There are 19 possible error messages in the disk drive test and their likely causes are explained in the manual. Each pass of the memory tests checks every address in RAM 520 times, including the space normally occupied by the diagnostic program itself. When an error occurs the address, expected data, and actual data are printed out together with a detailed error analysis showing the failing bit or bits, the corresponding IC's and their location. This is the most thorough test routine available for TRS-80 disk users.

BOOKS

LEVEL II ROM REFERENCE MANUAL $24.95 + $1.20 p&p

Over 70 pages packed full of useful information and sample programs. Applies to both TRS-80 and SYSTEM 80.

TRS-80 DISK AND OTHER MYSTERIES $24.95 + $1.20 p&p

The hottest selling TRS-80 book in the U.S.A. Disk file structures revealed, DOS's compared and explained, how to recover lost files, how to rebuild crashed directories – this is a must for the serious Disk user and is a perfect companion to any of the NEWDOS's.

LEARNING LEVEL II $16.95 + $1.20 p&p

Written by Daniel Lien, the author of the TRS-80 Level I Handbook, this book teaches you, step-by-step, how to get the most from your Level II machine. Invaluable supplement to either the TRS-80 Level II Manual or the System-80 Manuals.

HIGH QUALITY DISKETTES ALL PRICES INCLUDE P&P

40 TRACK NASHUA SINGLE SIDE/SINGLE DENSITY ... $45.00 box of 10

HIGH QUAL

40 TRACK VERBATIM DOUBLE SIDE

SOFTWARE BY AUSTRALIAN AUTHORS

All our software is suitable for either the SYSTEM 80 or the TRS-80

NEW SOFTWARE FROM MICRO-80 PRODUCTS

BUSINESS PROGRAMS

MICROMANAGEMENT STOCK RECORDING SYSTEM (L2/16K)

Cassette version. . . . . $29.95 + $1.00 p&p
Stringy Floppy version. . . . . $33.95 + $1.00 p&p

This system has been in use for 9 months in a number of small retail businesses in Adelaide. It is therefore thoroughly debugged and has been tailor made to suit the requirements of a small business. MICROMANAGEMENT SRC enables you to monitor the current stock level and reorder levels of 500 different stock items per tape or wafer. It includes the following features:—

- Add new items to inventory
- Delete discontinued items from inventory
- List complete file
- Search for any stock number
- Save data to cassette or wafer
- Load data from cassette or wafer
- Adjusts stock levels from sales results and receipt of goods
- List all items requiring reordering

We can thoroughly recommend this program for the small business with a L2/16K computer.

SCOTCH BRAND COMPUTING CASSETTES

Super-quality personal computing cassettes.
C-10 pack of 10 ... ... $26.00 incl. p&p
C-30 pack of 10 ... ... $28.00 incl. p&p

UTILITIES

S-KEY by Edwin Paay $15.95 plus 50c. p&p

S-KEY is a complete keyboard driver routine for the TRS-80 and becomes part of the Level II basic interpreter. With S-KEY loaded the user will have many new features not available with the standard machine.

S-KEY features:

* S-KEY provides an auto-repeat for all the keys on the keyboard. If any key is held down longer than about half a second, the key will repeat until it is released.
* Graphic symbols can be typed direct from the keyboard, this includes all 64 graphic symbols available from the TRS-80/SYSTEM 80.
* S-KEY allows text, BASIC commands and/or graphics to be defined to shifted keys. This makes programming much easier as whole commands and statements can be recalled by typing shift and a letter key.
* Because S-KEY allows graphics to be typed directly from the keyboard, animation and fast graphics are easily implemented by typing the appropriate graphics symbols directly into PRINT statements.
* S-KEY allows the user to LIST a program with PRINT statements containing graphics, properly. S-KEY does this by intercepting the LIST routine when necessary.
* S-KEY allows the user to list an updated list of the shift key entries to the video display or line printer.
* S-KEY can be disabled and enabled when required. This allows other routines which take control of the keyboard to run with S-KEY as well.

Each cassette has TRS-80, DISK and SYSTEM 80 versions and comes with comprehensive documentation.

BMON by Edwin Paay $19.95 plus 50c. p&p
THE ULTIMATE HIGH MEMORY BASIC MONITOR L2/16-48K

Our own personnel refuse to write BASIC without first loading this amazing machine language utility program into high memory! BMON Renumbers; Displays BASIC programs on the screen while they are still loading; tells you the memory locations of the program just loaded; lets you stop a load part-way through; merges two programs, with automatic renumbering of the second so as to prevent any clashes of line numbers; recovers your program even though you did type NEW: makes one program invisible while you work on a second (saves hours of cassette time!); lists all the variables used in the program; makes SYSTEM tapes; lets you Edit memory directly . . . the list goes on and on. Cassette comes with 16K, 32K and 48K versions, ready to load. Can anyone afford NOT to have BMON?

EDUCATIONAL

RPN CALCULATOR (L2/16K & 32K)
$14.95 $ 50c. p&p

Give your computer the power of a $650 reverse polish notation calculator with 45 functions and selectable accuracy of 8 or 16 digits. The main stack and registers are continuously displayed whilst the menu is always instantly accessible without disturbing any calculations or register values. The cassette comes with both the 16K and 32K versions, the latter giving you the additional power of a programmable calculator. Comes with a very comprehensive 15 page manual, which includes instructions to load and modify the 32K programmable version to run in 16K. Whether for business or pleasure, this package will prove invaluable, and turn you '80 into a very powerful instrument.

GAMES

MICROPOLY (L2/16K) $7.50 $ 50c p&p

Now you can play Monopoly on your micro. The old favourite board game has moved into the electronic era. This computer version displays the board on the screen, obeys all the rules and, best of all, the banker does not make mistakes with your change!

CONCENTRATION (L2/16K) $7.50 + 50c p&p

Another application of supergraphics. There are 28 "cards" displayed on the screen, face down. Players take it in turn to turn them over with the object of finding matching pairs. There are 40 different patterns which are chosen at random, so the game is full of endless variety. This is of particular value in helping young children to learn the art of concentrating and, at the same time, to introduce them to the computer.

METEOR AND TORPEDO ALLEY (L2/16K)
$9.95 + 50c p&p

Those who frequent games arcades will recognize these two electronic games. In METEOR you must destroy the enemy space ships before they see you. In its most difficult mode, the odds are a thumping 238 to 1 against you being successful. In torpedo alley you must sink the enemy ships without hitting your own supply ship. Both games include sound effects and are remarkably accurate reproductions of the arcade games.

TY DISKETTES

DOUBLE DENSITY ... $59.00 box of 10

HIGH QUALITY DISKETTES

77 TRACK VERBATIM SINGLE SIDE/SINGLE DENSITY ... $59.00 box of 10

AUSTRALIAN SOFTWARE (Cont.)

GAMES

SHEEPDOG (L2/16K) **$7.50 + 50c p&p**
Ever wondered how a sheepdog manages to drive all those awkward sheep into a pen? Well, here is your chance to find out just how difficult it is and have a lot of fun at the same time. You control the sheepdog, the computer controls the sheep! As if that isn't enough, look out for the dingoes lurking in the bush!

U BOAT **$7.50 plus 50c p&p**
Real time simulation at its best! Comes with working sonar-screen and periscope, a full rack of torpedoes, plenty of targets, working fuel and battery meters, helpful Mothership for high-seas reprovisioning and even has emergency radio for that terrible moment when the depth charges put your crew at risk. Requires Level II/16K.

SPACE INVADERS WITH SOUND **$7.50 + 50c p&p**
Much improved version of this arcade favourite with redesigned laser and cannon blasts, high-speed cannon, 50 roving drone targets, 10 motherships and heaps of fun for all. Level II with 4K and 16K versions on this cassette.

GOLF (L2/16K) **$7.50 + 50c p&p**
Pit your skills of mini-golf against the computer. Choose the level of difficulty, the number of holes and whether you want to play straight mini golf or crazy golf. Complete with hazards, water traps, bunkers and trees. Great fun for kids of all ages.

DOMINOES(L2/16K) **$7.50 + 50c p&p**
Pit your skill at dominoes against the computer, which provides a tireless opponent. Another application of supergraphics from the stable of Charlie Bartlett. Dominoes are shown approximately life size in full detail (except for colour!). The monitor screen is a window which you can move from one end of the string of dominoes to the other. Best of all, you don't lose any pieces between games!

KID'S STUFF (formerly MMM-1) $7.50 plus 50c. p&p
Three games on one cassette from that master of TRS-80 graphics, Charlie Bartlett. Includes INDY 500, an exciting road race that gets faster and faster the longer you play, SUBHUNT in which your warship blows up unfortunate little submarines all over the place, and KNIEVEL (as in motorcycle, ramp and buses).

OTHER PROGRAMS

INFINITE BASIC BY RACET (32K/1 DISK)
$49.95 + 50c. p&p
Full matrix functions – 30 BASIC commands; 50 more STRING functions as BASIC commands.

GSF/L2/48K **$24.95 + 50c. p&p**
18 machine language routines including RACET sorts.

BUSINESS ADDRESS AND INFORMATION SYSTEM (48K/DISK) **$24.95 + 50c. p&p**
Allows you to store addresses and information about businesses, edit them and print them out.

HISPED (L216, 32 or 48K) $29.95
This machine language program allows you to SAVE and LOAD programs and data to tape at speeds up to 2000 band (4 times normal) using a standard cassette recorder. A switch must be installed to remove the XRX III loading board, if fitted.

LOWER CASE FOR YOUR TRS-80/SYSTEM 80
Kit only $49.00 plus $2.00 p&p

Give your TRS-80 or SYSTEM 80 a lower case display with proper descenders and a block cursor (similar to the TRS-80 Model III). Also includes symbols for the four suits of cards. Includes full fitting instructions, all necessary components and a special machine language driver program to enable lower case in BASIC. The modification is similar to the Tandy model and does not work with Electric Pencil without further modifications.

These kits require disassembly of your computer and some soldering. They should only be installed by someone who has experience in soldering integrated circuits, using a low power, properly earthed soldering iron. If you do not have the necessary experience/equipment, we will install the modification for you for $20 plus freight in both directions. Make sure you arrange the installation with us first, before despatching your computer, so that we can assure you of a rapid turn-around. We are also arranging to have installers in each State. See elsewhere in this issue for their names and addresses.

PRICES

Cat No.
HD-020 Lower case mod kit for TRS-80
$49.00 plus $2.00 p&p
HD-021 Lower case mod kit for SYSTEM-80
$49.00 plus $2.00 p&p

EPSON MX-80 PRINTER
ONLY *$949 Inc. Cable for TRS-80 and p&p
(*Printer only – $940 incl. p&p)

The EPSON MX-80 printer is compact, quiet, has features unheard of only 2-3 years ago in a printer at any price and, above all, is ultra-reliable. All available print modes may be selected under software control. Features include:

- high quality 9x9 dot-matrix character formation
- 3 character densities
  - . 80 characters per line at 10 chars/inch
  - .132 characters per line at 16.5 chars/inch
  - . 40 characters per line at 5 chars/inch
- 2 line spacings
  - . 6 lines per inch . 8 lines per inch
- 80 characters per second print speed
- bi-directional printing
- logical seeking of shortest path for printing
- lower case with descenders
- TRS-80 graphics characters built in
- standard Centronics printer port

The bi-directional printing coupled with the logical seeking of the shortest print path (which means that the print head will commence printing the next line from the end which requires the least travel, thereby minimising unutilised time) gives this printer a much higher throughput rate than many other printers quoting print speeds of 120 c.p.s. or even higher.

GREEN SCREEN SIMULATOR
$19.95 incl. p&p

The GREEN SCREEN SIMULATOR is made from a deep green perspex, cut to fit your monitor. It improves contrast and is much more restful to the eyes than the normal grey and white image.

All editorial staff of MICRO-80 are now using GREEN SCREEN SIMULATORS on their own monitors.

Please make sure to specify whether you have an old (squarish) or new (rounded) style monitor when ordering. Not available for Dick Smith monitors.

```
6114 PRINT"SBC HL,SP";:RETURN
6115 PRINT"LD (";:GOSUB4600:PRINT"),SP";:RETURN
6120 PRINT"IN A,(C)";:RETURN
6121 PRINT"OUT (C),A";:RETURN
6122 PRINT"ADC HL,SP";:RETURN
6123 PRINT"LD SP,(";:GOSUB4600:PRINT")";:RETURN
6160 PRINT"LDI";:RETURN
6161 PRINT"CPI";:RETURN
6162 PRINT"INI";:RETURN
6163 PRINT"OUTI";:RETURN
6168 PRINT"LDD";:RETURN
6169 PRINT"CPD";:RETURN
6170 PRINT"IND";:RETURN
6171 PRINT"OUTD";:RETURN
6176 PRINT"LDIR";:RETURN
6177 PRINT"CPIR";:RETURN
6178 PRINT"INIR";:RETURN
6179 PRINT"OTIR";:RETURN
6184 PRINT"LDDR";:RETURN
6185 PRINT"CPDR";:RETURN
6186 PRINT"INDR";:RETURN
6187 PRINT"OTDR";:RETURN
7007 REM ** PRINT DD & FD INSTRUCTIONS **
7009 L=2:PRINT"ADD ";G$;",BC";:RETURN
7025 L=2:PRINT"ADD ";G$;",DE";:RETURN
7033 L=4:PRINT"LD ";G$;",";:GOSUB4600:RETURN
7034 L=4:PRINT"LD (";:GOSUB4600:PRINT"),";G$;:RETURN
7035 L=2:PRINT"INC ";G$;:RETURN
7041 L=2:PRINT"ADD ";G$;",";G$;:RETURN
7042 L=4:PRINT"LD ";G$;",(";:GOSUB4600:PRINT")";:RETURN
7043 L=2:PRINT"DEC ";G$;:RETURN
7052 L=3:PRINT"INC ";P$;:RETURN
7053 L=3:PRINT"DEC ";P$;:RETURN
7054 L=4:PRINT"LD ";P$;",";:GOSUB4700:RETURN
7057 L=2:PRINT"ADD ";G$;",SP";:RETURN
7070 L=3:PRINT"LD B,";P$;:RETURN
7078 L=3:PRINT"LD C,";P$;:RETURN
7086 L=3:PRINT"LD D,";P$;:RETURN
7094 L=3:PRINT"LD E,";P$;:RETURN
7102 L=3:PRINT"LD H,";P$;:RETURN
7110 L=3:PRINT"LD L,";P$;:RETURN
7112 L=3:PRINT"LD ";P$;",B";:RETURN
7113 L=3:PRINT"LD ";P$;",C";:RETURN
7114 L=3:PRINT"LD ";P$;",D";:RETURN
7115 L=3:PRINT"LD ";P$;",E";:RETURN
7116 L=3:PRINT"LD ";P$;",H";:RETURN
7117 L=3:PRINT"LD ";P$;",L";:RETURN
7119 L=3:PRINT"LD ";P$;",A";:RETURN
7126 L=3:PRINT"LD A,";P$;:RETURN
7134 L=3:PRINT"ADD A,";P$;:RETURN
7142 L=3:PRINT"ADC A,";P$;:RETURN
7150 L=3:PRINT"SUB ";P$;:RETURN
7158 L=3:PRINT"SBC A,";P$;:RETURN
7166 L=3:PRINT"AND ";P$;:RETURN
7174 L=3:PRINT"XOR ";P$;:RETURN
7182 L=3:PRINT"OR ";P$;:RETURN
7190 L=3:PRINT"CP ";P$;:RETURN
7225 L=2:PRINT"POP ";G$;:RETURN
7227 L=2:PRINT"EX (SP),";G$;:RETURN
7229 L=2:PRINT"PUSH ";G$;:RETURN
7233 L=2:PRINT"JP (";G$;")";:RETURN
7249 L=2:PRINT"LD SP,";G$;:RETURN
7999 REM ** 8000 - ED INSTRUCT.
8000 GOSUB11000:Y$=RIGHT$(STR$(F+6000),4):GOSUB2110
8005 IFF=67ORF=75ORF=83ORF=91ORF=115ORF=123THENL=4ELSEL=2
8010 M=17231:GOSUB2150:GOTO50
9999 REM ** PRINT CURRENT MEMORY CONTENTS (ALL INSTR.) **
10000 W=0:FORT=1TOL:E=PEEK(Z-1+T):GOSUB5000
10010 PRINT@15+(K-1)*64+W,C$;:W=W+2:NEXT
10011 REM ** PRINT ASCII CHARACHTERS **
10012 W=0:FORT=1TOL:S=PEEK(Z-1+T):IFNOT(S>64ANDS<91)THENS=46
10016 PRINT@54+(K-1)*64+W,CHR$(S);:W=W+2:NEXT:PRINT:RETURN
10019 REM ** CB INSTRUCTIONS **
10020 L=2:N=PEEK(Z+1):IFN>=64THENGOTO10180
10030 GOSUB10050:GOSUB10150
10040 PRINTQ$;" ";D$;:RETURN
```

```
10050 IFN<8THENQ$="RLC":RETURN
10060 IFN<16THENQ$="RRC":RETURN
10070 IFN<24THENQ$="RL":RETURN
10080 IFN<32THENQ$="RR":RETURN
10090 IFN<40THENQ$="SLA":RETURN
10100 IFN<56THENQ$="SRA":RETURN
10110 IFN<64THENQ$="SRL":RETURN
10120 IFN<128THENQ$="BIT":RETURN
10130 IFN<192THENQ$="RES":RETURN
10140 Q$="SET":RETURN
10150 R=N/8:R=N-R*8+1
10160 RESTORE:FORT=1TOR:READD$:NEXT:RETURN
10170 DATAB,C,D,E,H,L,(HL),A
10180 GOSUB10120:GOSUB10150:GOSUB10200
10190 PRINTQ$;J;",";D$;:RETURN
10200 IFN>127ANDN<192THENO=128:GOTO10220
10210 IFN>191THENO=192ELSEO=64
10220 J=(N-O)/8:RETURN
10999 REM ** PEEK AT 2ND BYTE OF DD/ED/FD INSTR. **
11000 F=PEEK(Z+1):IFNOTF=203THENRETURNELSEL=4
11005 N=PEEK(Z+3):IFN<63THEN11100ELSEU=N:U=U/8
11010 IFU>7THENU=U-8:GOTO11010
11020 GOSUB10120:PRINTQ$;U;",";P$;:GOTO30
11100 GOSUB10050:PRINTQ$;" ";P$;:GOTO30
12000 IFERR/2+1=8THENPRINT"? DATA ?";:L=1:RESUME30
12002 PRINT:PRINT"ERROR NUMBER";ERR/2+1;"IN LINE";ERL:END
```

***** 12 HOUR CLOCK L2/4K m/l (c) Brenton Hoff *****

Here is a clock program which is ideally suited to loading into the '80 when nothing else is being run on it. It features large digits and high accuracy (the time is displayed to the hundredth of a second). Operation is easy and fast and the program is designed to minimise interference caused on the screen.

ENTERING THE PROGRAM

Using a suitable monitor such as BMON, RSM etc., enter the Hex listing shown below, into memory. (TBUG is not suitable as it resides in the same area as CLOCK). Punch out a SYSTEM tape using the following parameters:-

START	END	ENTRY	NAME
4A36	4BA0	4A36	CLOCK

LOADING FROM CASSETTE

1. Switch on the computer
2. Answer MEMORY SIZE?/READY? with ENTER/NEWLINE.
3. Type SYSTEM and press ENTER/NEWLINE
4. Answer *? by typing CLOCK, press ENTER/NEWLINE
5. When the program has loaded, answer the second *? by typing / and press ENTER/NEWLINE to begin execution.

The program will immediately display:-

HHMMSSHH?

where the first two H's stand for hours, the two M's for minutes, the two S's for seconds and the last two H's for hundredths of seconds. You must enter all 8 digits. E.g. 4:10 would be entered as 04100000. Type in a time which is a little ahead of the actual time and then press the ENTER/NEWLINE key at the instant the time is correct. For extreme accuracy, add an extra 0.10 seconds to the time at which you intend to press the ENTER/NEWLINE key. This allows for the input processing time of the computer.

The clock will commence functioning at the time typed in and requires no further input. The time is displayed in large graphics and, at least on the TRS-80 monitor, the interference caused by the clock display often synchronises quite accurately with the electron beam, the speed and direction of the interference being closely related to the calibration of the clock.

```
4A36:  21 F7 4A CD A7 28 CD B3 1B CD 78 1D 11 E5 4A 06
4A46:  08 7E D6 30 FA 36 4A FE 3A 30 E5 12 23 13 10 F1
4A56:  CD C9 01 06 07 21 E5 4A 7E E5 21 F0 4A B7 28 04
4A66:  36 02 18 02 36 09 E1 3E B3 32 12 3E 32 20 3E 3E
4A76:  B0 32 2E 3E 04 B7 7E E5 68 2D 26 4B 5E 16 3D 4F
4A86:  17 17 17 17 C6 08 91 6F 3E 03 C5 01 05 00 ED B0
4A96:  01 3B 00 EB 09 EB 3D 20 F2 C1 E1 23 10 D7 01 8B
4AA6:  02 CD 60 00 00 00 00 00 21 EC 4A 11 F6 4A 01 00
4AB6:  00 34 1A BE 30 06 71 2B 1B 04 18 F5 78 FE 07 38
4AC6:  B3 E5 21 F0 4A 3E 0B AE 77 21 E5 4A 7E 23 B6 20
4AD6:  02 36 01 E1 78 FE 08 20 9B 23 05 18 97 00 00 00
4AE6:  00 00 00 00 00 00 00 FF FF 01 09 05 09 05 09 09
4AF6:  09 48 48 4D 4D 53 53 68 68 00 F6 F0 E8 E2 DA D4
4B06:  CC C6 BC 83 83 B3 BC BF 8C 83 80 BF 80 83 83 83
4B16:  80 80 8C BF 80 80 80 80 BF 80 80 80 83 83 83 80
4B26:  8C 83 83 83 BC BC 83 83 83 80 83 83 83 83 83 8C
4B36:  83 83 83 BC B0 80 83 83 BC 80 83 83 83 80 80 B0
4B46:  8C BF 80 8F 8C 8C BF 8C 80 80 80 83 80 BF B3 B3
4B56:  B3 83 B0 80 80 80 BF 80 83 83 83 80 B0 8C 83 83
4B66:  80 BF 83 83 83 BC 80 83 83 83 80 83 83 83 B3 8F
4B76:  B0 8C 83 80 80 83 80 80 80 80 BC 83 83 83 BC BC
4B86:  83 83 83 BC 80 83 83 83 80 BC 83 83 83 BC 80 83
4B96:  83 B3 8F 80 83 83 80 80 47 FF FF
```

***** BONES LII/4K (c) David Balaic *****

LOADING INSTRUCTIONS.

To load the program from cassette, answer MEMORY SIZE? (READY?) by pressing ENTER/NEW LINE. Then type CLOAD"B".
When the program has loaded type RUN and press ENTER/NEWLINE.

HOW TO PLAY.

The object of this reflex game for two players is to be the first to build a complete skeleton. The screen is divided down the middle, the first player's skeleton will be built on the left of the screen and the second on the right. The left hand player uses the "Z" key and the right hand player uses the "/" key.

At the bottom of the screen the words:

PRESS A KEY --

will be displayed. When the word "NOW" appears the first player to press his key will get some bones, but be careful. If you press your key before the word "NOW" appears you lose ALL your bones.

```
10          'BONES BY DAVID BALAIC, 194/125 NAPIER ST., FITZROY 3065
                  COPYRIGHT (C)          COMPLETED 4/11/1980.
20 CLS:PRINT@221,"BONES";:RANDOM
30 FORX=59TO67:SET(X,23):NEXT:SET(59,27):SET(67,27):SET(63,26):SET(60,25):SET(61
,25):SET(65,25):SET(66,25):FORY=24TO26:SET(58,Y):SET(68,Y):NEXT
40 FORX=60TO66STEP2:SET(X,28):RESET(X,30):SET(X,29):NEXT
50 FORX=60TO66STEP2:RESET(X,29):SET(X,30):NEXT:Z=Z+1
60 IFZ=27THEN70ELSEFORN=1TO100:NEXT:GOTO40
70 CLS:PRINT"              THE BONES PROGRAM BY DAVID BALAIC":PRINT
80 PRINT"THIS GAME TESTS YOUR REFLEXES. YOU MUST BEAT YOUR OPPONENT BY
PRESSING YOUR KEY (Z FOR PLAYER #1, / FOR PLAYER #2) WHEN TOLD."
90 INPUT"HIT 'ENTER' TO CONTINUE";Z$
100 CLS:FORY=0TO44:SET(61,Y):SET(62,Y):NEXT:G=28:M=32:J=63
110 C$=INKEY$:PRINT@960,"PRESS THE BUTTON. . .                    ";:FORN=0TORND(200
0):B$=INKEY$:IFB$<>""THEN130ELSENEXT
120 GOTO160
130 IFB$="Z"PRINT@458,"CHEAT!";:GOSUB400
140 IFB$="/"PRINT@490,"CHEAT!";:GOSUB400
150 NEXT
160 PRINT@983,"N O W";:A$=INKEY$:IFA$=""THEN160ELSE170
170 IFA$="Z"T=T+1:GOTO210
180 IFA$="/"S=S+1:GOTO300
190 IF(A$="Z/")OR(A$="/Z")A$=LEFT$(A$,1):GOTO170ELSE200
200 PRINT@960,"DRIPS! -PRESS 'Z' OR '/'! COR BLIMEY!";:FORN=1TO1300:NEXT:GOTO110
210 IFT=1FORX=11TO21:SET(X,44):NEXT:FORY=29TO43:SET(21,Y):NEXT:GOTO110
220 IFT=2FORX=30TO40:SET(X,44):NEXT:FORY=29TO43:SET(30,Y):NEXT:GOTO110
230 IFT=3FORX=19TO32:SET(X,24):SET(X,28):NEXT:FORY=10TO23:SET(25,Y):SET(26,Y):NE
XT:SET(21,26):SET(30,26):FORY=25TO27:SET(19,Y):SET(32,Y):NEXT:GOTO110
```

```
240 IFT=4FORX=15TO36:SET(X,9):NEXT:FORX=23TO28:SET(X,3):NEXT:FORY=4TO5:SET(22,Y)
:SET(29,Y):NEXT:SET(23,6):SET(28,6):SET(27,5):SET(24,5):FORX=24TO27:SET(X,7):NEX
T:SET(25,8):SET(26,8):GOTO110
250 IFT=5FORY=10TO20:SET(15,Y):NEXT:SET(15,29):FORY=21TO28:SET(14,Y):SET(16,Y):N
EXT:FORY=30TO32:SET(14,Y):SET(15,Y):SET(16,Y):NEXT:SET(13,31):SET(17,31):SET(15,
33):GOTO110
260 IFT=6FORY=10TO20:SET(36,Y):NEXT:SET(36,29):FORY=21TO28:SET(35,Y):SET(37,Y):N
EXT:FORY=30TO32:SET(35,Y):SET(36,Y):SET(37,Y):NEXT:SET(34,31):SET(38,31):SET(36,
33):GOTO110
270 IFT=7FORX=18TO33:SET(X,11):SET(X,13):NEXT:GOTO110
280 IFT=8FORX=18TO33:SET(X,16):SET(X,18):SET(X,20):NEXT:GOTO110
290 IFT=9PRINT@5,"PLAYER #1 WINS";:PRINT@36,"DO YOU WISH TO PLAY AGAIN";:INPUTZ$
:Z$=LEFT$(Z$,1):IFZ$="Y"THENRUN100ELSEEND
300 IFS=1FORX=74TO84:SET(X,44):NEXT:FORY=28TO43:SET(84,Y):NEXT:GOTO110
310 IFS=2FORX=93TO103:SET(X,44):NEXT:FORY=28TO43:SET(93,Y):NEXT:GOTO110
320 IFS=3FORX=82TO95:SET(X,24):SET(X,28):NEXT:FORY=25TO27:SET(82,Y):SET(95,Y):NE
XT:SET(84,26):SET(93,26):FORY=10TO23:SET(88,Y):SET(89,Y):NEXT:GOTO110
330 IFS=4FORX=78TO99:SET(X,9):NEXT:FORX=87TO90:SET(X,7):NEXT:SET(88,8):SET(89,8)
:SET(87,5):SET(90,5):SET(91,6):SET(86,6):FORY=4TO5:SET(85,Y):SET(92,Y):NEXT:FORX
=86TO91:SET(X,3):NEXT:GOTO110
340 IFS=5FORY=10TO20:SET(78,Y):NEXT:FORY=21TO28:SET(77,Y):SET(79,Y):NEXT:FORY=30
TO32:SET(77,Y):SET(78,Y):SET(79,Y):NEXT:SET(78,33):SET(78,29):SET(76,31):SET(80,
31):GOTO110
350 IFS=6FORY=10TO20:SET(99,Y):NEXT:SET(99,29):FORY=21TO28:SET(98,Y):SET(100,Y):
NEXT:FORY=30TO32:SET(98,Y):SET(99,Y):SET(100,Y):NEXT:SET(99,33):SET(97,31):SET(1
01,31):GOTO110
360 IFS=7FORX=81TO96:SET(X,11):SET(X,13):NEXT:GOTO110
370 IFS=8FORX=81TO96:SET(X,16):SET(X,18):SET(X,20):NEXT:GOTO110
380 IFS=9PRINT@36,"PLAYER #2 WINS";:PRINT@5,"DO YOU WISH TO PLAY AGAIN";:INPUTZ$
:Z$=LEFT$(Z$,1):IFZ$="Y"THENRUN100ELSEEND
390 END
400 IFB$="/"THEN450
410 FORN=1TO500:NEXT
420 FORE=FTOG:IFE=924THEN440ELSEPRINT@E," ";:NEXT
430 F=F+64:G=G+64:GOTO420
440 F=0:G=28:T=0:B$="":GOTO110
450 FORN=ITO500:NEXT
460 FORH=MTOJ:IFH=959THEN480ELSEPRINT@H," ";:NEXT
470 M=M+64:J=J+64:GOTO460
480 M=32:J=63:S=0:B$="":GOTO110
```

***** PHILATELIC ADVISOR LII/16K (c) John S. Richardson *****

Here is a program for those of our readers who are collectors or investors, specifically of stamps but the same principles could be applied to many other fields.

This program uses most of the 16K of memory due to the large data base. It would be simple to list other stamps by changing the data and relevant titles. The last two items of each data statement refer to the author's stamp collection. They should be amended to suit the user's own collection. If additional sets are added or deleted, the value of N should be changed to reflect the revised number of sets. The program holds data on each stamp set issued in Australia from 1966 (decimal currency) to the end of 1980. The data includes the set name, year/issue, face value, number of stamps in the set and the 1980 catalogue value. It also calculates the annual compound growth rate, (not just the simple growth rate), of each stamp, catalogue/face value. It sorts into sequence for growth, i.e. stamps with the highest growth listed first; this is a long sort that takes about 7 minutes. Finally, the program lets the user input current values and calculates growth and sorts into sequence. This program should appeal to the many stamp investors. Australia Post estimates that there are over 1 million collectors in Australia, 300,000 of whom collect Australian decimal stamps.

```
10 GOTO40
20 INPUT"PRESS 'NEW LINE' TO CONTINUE";C$
30 RETURN
40 CLEAR 10
50 D=80:CLS
60 N=106
70 REM:N=NUMBER OF STAMP SETS TO END 'D'
80 T$="##.##":U$="###.##":V$="$$###.##"
90 DIMS$(N):DIMS(N,7):DIMT(N):DIME(N):DIMG(N)
100 PRINT@514,"** STAMPS **  STAMPS  **  STAMPS  **  STAMPS  **  STAMPS  **
```

```
110 FORX=1TO500:NEXT
120 PRINT@208,"PHILATELIC ADVISOR"
130 PRINTTAB(13)"AND COLLECTION EVALUATOR"
140 PRINT:PRINTTAB(13)"AUSTRALIA - DECIMAL SETS"
150 PRINT@776,"A PROGRAMME BY JOHN S RICHARDSON"
160 FORX=1TO2000:NEXT X:CLS
170 PRINT"THIS PROGRAMME ALLOWS THE USER TO REVIEW THE VALUE OF SETS OF
AUSTRALIAN STAMPS, SINCE DECIMALISATION IN 1966."
180 PRINT"DATA IS CURRENTLY VALID FOR ALL ISSUES TO END '"D"."
190 PRINT:PRINT"CAT. VALUES REFER TO THE AUSTRALIAN STAMP CATALOGUE,
"D+1"EDITION, PRINTED MID '"D". THE USER MAY ADD CURRENT VALUES
TO OBTAIN UP-TO-DATE DATA ON RECENT GROWTH."
200 PRINT:PRINT "DATA HELD IS FOR j.RICHARDSON'S STAMP COLLECTION AT END '"D
210 PRINT"FOR RECORD AND VALUATION PURPOSES."
220 PRINT:PRINT"IMPORTANT:ALL STAMPS MINT"
230 PRINT
240 GOSUB20
250 FORX=1TON
260 READS(X,1)
270 READS$(X)
280 FORY=2TO6
290 READS(X,Y)
300 NEXTY
310 NEXTX
320 GOTO1960
330 ONRGOTO340,590,1040,1040,1040,1720
340 CLS:PRINTTAB(15)"REPORT NO. 1"
350 PRINT:PRINT"THIS REPORT WILL PROVIDE DETAILS OF ALL AUSTRALIAN STAMP SETS
SINCE DECIMALISATION."
360 PRINT"IT WILL ALSO SHOW THE NUMBER OF SETS PURCHASED FOR THE
COLLECTION AT END '"D"."
370 PRINT"EACH SET WILL BE VALUED ACCORDING TO A.S.C."D+1", AND TOTALS
WILL BE PRINTED.
380 GOSUB20
390 Z=0:TA=0:TB=0:TY=0:TX=0
400 GOSUB960
410 FORX=1TON
420 TA=S(X,4)*S(X,5)
430 TX=TX+TA
440 TB=S(X,5)*S(X,6)
450 TY=TY+TB
460 PRINTS(X,1)TAB(5)S$(X)TAB(15)S(X,2);
470 PRINTTAB(20)USINGT$;S(X,3);
480 PRINTTAB(28)USINGU$;S(X,4);
490 PRINTTAB(38)S(X,5);
500 PRINTTAB(46)USINGU$;S(X,6)
510 Z=Z+1
520 IFZ=13THENGOSUB20
530 IFZ=13THENGOSUB960
540 NEXT X
550 PRINT:PRINT"TOTAL VALUE OF COLLECTION IS $"TX"AT CAT. PRICE"
560 PRINT"TOTAL COST OF COLLECTION WAS $"TY
570 GOSUB20
580 GOTO1960
590 CLS
600 IFI=1GOTO750
610 IFR>3 GOTO650
620 PRINTTAB(16)"REPORT NO. 2":PRINT
630 PRINT"THIS REPORT LISTS ALL AUSTRALIAN STAMP SETS AT CURRENT VALUES
AND PROVIDES DATA ON THE COLLECTION. TOTALS ARE PRINTED."
640 PRINT:FOR X=1TO500:NEXT
650 PRINT"THIS REPORT UTILISES CURRENT VALUES. YOU WILL BE REQUIRED
TO ENTER A VALUE FOR EACH STAMP SET."
660 PRINT"PREPARE TO ENTER VALUES....."
670 FORX=1TON
680 PRINT:PRINT"ENTER CURRENT VALUE OF..."S(X,1) S$(X)" ("S(X,2)"). CAT";
690 PRINTUSINGV$;S(X,4);
700 INPUTS(X,7)
710 NEXTX
720 I=1
730 IFR=4GOTO1080
740 IFR=5GOTO1080
750 Z=0:TA=0:TB=0:TX=0:TY=0
760 GOSUB1000
770 FORX=1TON
780 TA=S(X,5)*S(X,7)
```

```
790 TX=TX+TA
800 TB=S(X,6)*S(X,5)
810 TY=TY+TB
820 PRINT S(X,1)TAB(5)S$(X)TAB(15)S(X,2);
830 PRINTTAB(19)USINGT$;S(X,3);
840 PRINTTAB(26)USINGU$;S(X,4);
850 PRINTTAB(34)USINGU$;S(X,7);
860 PRINTTAB(43)S(X,5);
870 PRINTTAB(51)USINGU$;S(X,6)
880 Z=Z+1
890 IFZ=13GOSUB20
900 IFZ=13GOSUB1000
910 NEXTX
920 PRINT:PRINT"TOTAL VALUE OF COLLECTION AT CURRENT VALUE IS $"TX
930 PRINT"TOTAL COST OF COLLECTION WAS $"TY
940 GOSUB20
950 GOTO1960
960 CLS:PRINTTAB(1)"YR"TAB(6)"DESC"TAB(15)"NO"TAB(20)"FACE$"TAB(28)"CAT$"TAB(35)
"QTY P/D"TAB(46)"COST/SET"
970 PRINT"-------------------------------------------------------------"
980 Z=0
990 RETURN
1000 CLS:PRINTTAB(1)"YR"TAB(6)"DESC"TAB(16)"NO"TAB(20)"FACE$"TAB(27)"CAT$"TAB(34
)"CURR$"TAB(41)"QTY P/D"TAB(50)"COST/SET"
1010 PRINT"--------------------------------------------------------------"
1020 Z=0
1030 RETURN
1040 REM:RANKING REPORTS
1050 IFI=1GOTO1080
1060 IFR=4GOTO590
1070 IFR=5GOTO590
1080 IFR=3GOTO1130
1090 IFR=4GOTO1160
1100 CLS:PRINTTAB(16)"REPORT NO. 5"
1110 PRINT:PRINT"THIS REPORT WILL RANK EACH STAMP SET IN SEQUENCE, BASED ON
THE GROWTH IN ITS CURRENT VALUE, COMPARED TO CAT. VALUE. IT IS
A GOOD GUIDE TO SHORT TERM, RECENT GROWTH."
1120 GOTO1180
1130 CLS:PRINTTAB(16)"REPORT NO. 3"
1140 PRINT:PRINT"THIS REPORT WILL RANK EACH STAMP SET IN SEQUENCE, BASED ON
ITS INCREASE IN VALUE PER ANNUM, CAT COMPARED TO FACE VALUE."
1150 GOTO1180
1160 CLS:PRINTTAB(16)"REPORT NO. 4"
1170 PRINT:PRINT"THIS REPORT WILL RANK EACH STAMP SET IN SEQUENCE, BASED ON
ITS INCREASE IN VALUE PER ANNUM, CURRENT COMPARED TO FACE VALUE."
1180 PRINT:GOSUB20
1190 CLS:PRINT@454,"HMMMMMMM.....LET ME THINK....."
1200 PRINT@582,"(THIS IS A COMPLEX CALCULATION - ALLOW UP TO 10 MINS.)"
1210 FORX=1TON
1220 G(X)=0
1230 T(X)=D-S(X,1)
1240 IFT(X)<1THENT(X)=1
1250 T(X)=1/T(X)
1260 IFR=4THEN1290
1270 IFR=5THEN1300
1280 G(X)=(((S(X,4)/S(X,3))[T(X))-1)*100:GOTO 1310
1290 G(X)=(((S(X,7)/S(X,3))[T(X))-1)*100:GOTO 1310
1300 G(X)=((S(X,7)/S(X,4))-1)*100
1310 G(X)=INT(G(X)*10)/10
1320 NEXTX
1330 GOSUB1620
1340 GOSUB1500
1350 FORX=1TON
1360 Z=Z+1
1370 PRINTXTAB(4)S(E(X),1)TAB(9)S$(E(X))TAB(19)S(E(X),2);
1380 PRINTTAB(24)USINGT$;S(E(X),3);
1390 PRINTTAB(31)USINGU$;S(E(X),4);
1400 PRINTTAB(40)USINGU$;S(E(X),7);
1410 PRINTTAB(50)G(E(X))TAB(59)S(E(X),5)
1420 IFZ<12GOTO1450
1430 GOSUB20
1440 GOSUB 1500
1450 NEXTX
1460 GOSUB20
1470 INPUT"LIKE TO VIEW THIS DISPLAY AGAIN (Y/N)";A$
1480 IFA$="Y"THEN1340
1490 GOTO 1960
```

```
1500 CLS:PRINT"RNK"TAB(5)"YR"TAB(9)"DESC"TAB(19)"NO"TAB(24)"FACE$"TAB(32)"CAT$"T
AB(39)"CURR$";
1510 IF R=4 GOTO 1550
1520 IF R=5 GOTO 1580
1530 PRINTTAB(47)"CAT/FACE %"TAB(59)"QTY"
1540 GOTO1590
1550 PRINTTAB(46)"CURR/FACE %"TAB(59)"QTY"
1560 GOTO1590
1570 RETURN
1580 PRINTTAB(47)"CURR/CAT %"TAB(59)"QTY"
1590 Z=0
1600 PRINT"------------------------------------------------------------------"
1610 RETURN
1620 A=0:B=0:P=0
1630 FORA=1TON
1640 P=1
1650 FORB=1TON
1660 IFG(A)<G(B)THENP=P+1
1670 IFG(A)=G(B)ANDA<BTHENP=P+1
1680 NEXTB
1690 E(P)=A
1700 NEXTA
1710 RETURN
1720 A=66
1730 GOSUB1920
1740 FORX=1TON
1750 IFS(X,1)=ATHEN1770
1760 GOTO1830
1770 TA=TA+S(X,3)
1780 TB=TB+S(X,4)
1790 B=D-A
1800 IFD-A=0THENB=1
1810 TC=((TB/TA)[(1/B))-1
1820 TC=INT(TC*1000)/10
1830 NEXTX
1840 PRINTATAB(10)TATAB(20)TBTAB(30)TC
1850 Z=Z+1
1860 IFZ=13GOSUB20
1870 IFZ=13GOSUB1920
1880 A=A+1:TA=0:TB=0:TC=0
1890 IFA=D+1GOSUB20
1900 IFA=D+1GOTO1960
1910 GOTO1740
1920 CLS:PRINT TAB(1)"YR"TAB(10)"FACE$"TAB(20)"CAT$"TAB(30)"GROWTH % P.A."
1930 PRINT"------------------------------------------------------------"
1940 Z=0
1950 RETURN
1960 CLS
1970 PRINTTAB(20)"** MENU **"
1980 PRINT:PRINT"YOU MAY SELECT FROM ONE OF THE FOLLOWING REPORTS:"
1990 PRINT"1. COLLECTION VALUATION BY CAT. VALUE."
2000 PRINT"2. COLLECTION VALUATION BY CURRENT VAL.      NOTE A"
2010 PRINT"3. RANKING, GROWTH P/A - CAT$/FACE$.
2020 PRINT"4. RANKING, GROWTH P/A - CURR$/FACE$.        NOTE A"
2030 PRINT"5. RANKING, SHORT-TERM GROWTH% - CUR/CAT.    NOTE A"
2040 PRINT"6. SUMMARY OF VALUES BY YEAR OF ISSUE."
2050 PRINT:PRINT"NOTE A: ENTER CURRENT VALUES OF ALL STAMP SETS TO END '"D
2060 PRINT:INPUT"ENTER REPORT NUMBER REQUIRED";R
2070 IFR<1ORR>8GOTO2060
2080 GOTO330
2090 DATA 66,QEii,7,.28,10.3,1,4.8
2100 DATA 66,QE COIL,3,.12,3.5,1,1.9
2110 DATA 66,BIRDS,8,1.38,130.25,0,0
2120 DATA 66,NAVS,6,8.65,243.5,0,0
2130 DATA 66,FISH,4,.34,34,1,19.8
2140 DATA 66,4C COMM'S,3,.12,1.2,4,.6
2150 DATA 67,4C COMM'S,5,.20,2,4,1
2160 DATA 67,5C ON 4C,1,.05,1.75,4,1.25
2170 DATA 67,xMAS,2,.3,9,4,5.7
2180 DATA 68,WWW,2,.25,10.5,2,8
2190 DATA 68,INTEL,1,.25,10,0,0
2200 DATA 68,FLOWERS,6,1.09,45.75,0,0
2210 DATA 68,SOIL/MED,2,.1,1.2,4,.4
2220 DATA 68,OLYMPIC,2,.3,8.5,0,0
2230 DATA 68,5C COMM'S,3,.15,1.5,4,.6
2240 DATA 68,FA'S 1,4,.2,12,0,0
2250 DATA 69,5C COMM'S,3,.15,1.5,4,.6
```

```
2260 DATA 69,PRY IND,4,.67,35,0,0
2270 DATA 69,XMAS,2,.3,10.5,0,0
2280 DATA 69,PM'S 1,4,.2,10,0,0
2290 DATA 69,FLIGHT ST,3,.15,15,0,0
2300 DATA 70,RAILWAY,1,.05,.5,4,.2
2310 DATA 70,EXPO,2,.25,6.5,0,0
2320 DATA 70,ROYAL V,2,.35,10.5,0,0
2330 DATA 70,GRASS,1,.05,.4,4,.2
2340 DATA 70,COOK BIC,6,.55,14,0,0
2350 DATA 70,COOK M/S,1,.55,20,0,0
2360 DATA 70,FLR COIL,6,.34,3.9,4,2.24
2370 DATA 70,NAT DEV,4,.34,13.6,0,0
2380 DATA 70,6C COMM,4,.24,2,4,.8
2390 DATA 70,QANTAS,2,.36,5.5,4,3.8
2400 DATA 70,FA'S,4,.24,10,0,0
2410 DATA 71,a'ASIA,3,.42,16.5,0,0
2420 DATA 71,6C COMM'S,4,.24,2,4,.8
2430 DATA 71,RSPCA,4,.62,11,0,0
2440 DATA 71,AB ART,4,1.1,5.35,0,0
2450 DATA 71,XMAS BLK,7,.49,125,0,0
2460 DATA 72,7C COMM'S,3,.21,1.5,4,.6
2470 DATA 72,FA'S,4,.28,8,0,0
2480 DATA 72,PRI IND,4,1.1,68,0,0
2490 DATA 72,REHAB,3,.54,6,1,3
2500 DATA 72,OLYMPIC,4,.56,12.7,0,0
2510 DATA 72,PIONEER,7,2.6,16.85,0,0
2520 DATA 72,XMAS,2,.42,18.4,1,16.25
2530 DATA 73,METRIC,4,.28,14,1,9.5
2540 DATA 73,7C COMM'S,3,.21,1.5,4,.65
2550 DATA 73,FA'S BLK,4,.28,18,1,13.5
2560 DATA 73,NAT DEV,4,1.1,52.5,1,30
2570 DATA 73,MAR GEM,9,.5,1.8,6,.5
2580 DATA 73,XMAS,2,.37,10.6,1,7.5
2590 DATA 73,ARCH,4,1.07,15.5,1,10
2600 DATA 74,ANIMALS,4,1.5,8.15,4,4.5
2610 DATA 74,7C COMM'S,2,.14,1.2,4,.5
2620 DATA 74,PAINTING,5,22,34,2,22
2630 DATA 74,SPORTS,7,.49,7,1,3.45
2640 DATA 74,UPU,2,.37,3.7,4,2.05
2650 DATA 74,9C SURCH,1,.09,.35,1,.25
2660 DATA 74,XMAS,2,.45,2.4,1,1.45
2670 DATA 74,EDUCAT,4,.91,6.3,1,5.35
2680 DATA 75,ENVIR'T,3,.3,3,2,2.5
2690 DATA 75,IWY,1,.1,.5,4,.2
2700 DATA 75,PM'S,6,.6,3.6,1,2.25
2710 DATA 75,SCIENCE,4,1.26,10,1,5
2720 DATA 75,POST/TEL,2,.2,3,2,2
2730 DATA 75,FA'S,6,.6,6,1,5.1
2740 DATA 75,W/FLRS,2,.63,1.5,4,.73
2750 DATA 75,PNG,2,.43,2.25,4,1.7
2760 DATA 75,XMAS,2,.6,4,1,2.05
2770 DATA 76,NAT'HD,2,.36,2.15,2,1.3
2780 DATA 76,18C COMMS,3,.54,1.5,4,.86
2790 DATA 76,EXPLORE,6,1.08,4.5,1,2.35
2800 DATA 76,OLYMPIC,4,1.01,2.95,1,1.6
2810 DATA 76,SCENES,6,2.7,6.4,5,3.94
2820 DATA 76,S/WK M/S,1,.72,3,1,2.95
2830 DATA 76,XMAS,2,.6,2,4,1.25
2840 DATA 77,FA'S,4,.72,2,1,1.2
2850 DATA 77,P.ARTS,4,1.5,3.9,4,2.25
2860 DATA 77,JUBILEE,2,.63,2.25,4,2.5
2870 DATA 77,CRICKET,6,1.35,6.5,1,4.4
2880 DATA 77,XMAS,2,.6,1.35,4,1.12
2890 DATA 78,18C COMMS,3,.54,1.1,4,.86
2900 DATA 78,AVIATORS,4,.72,1.6,1,1.4
2910 DATA 78,AVIAT M/S,1,.72,2,1,1.4
2920 DATA 78,TREES,4,1.28,5.75,1,4
2930 DATA 78,BIRDS 1,5,1.35,2.70,4,1.35
2940 DATA 78,S.WK,1,.2,.4,2,.3
2950 DATA 78,S.WK M/S,4,.8,2,1,1.5
2960 DATA 78,XMAS,3,.95,3.4,1,2.05
2970 DATA 78,HORSE R.,4,1.6,4.15,4,2.65
2980 DATA 79,20C COMMS,3,.6,1.2,4,.6
2990 DATA 79,FERRIES,4,1.6,2.2,4,1.6
3000 DATA 79,LOCOS,4,1.6,2.2,4,1.6
3010 DATA 79,N.PK,7,1.4,3.4,10,1.94
```

```
3020 DATA 79,BIRDS 2,6,1.28,2.6,4,1.28
3030 DATA 79,FISHING,4,1.6,3.2,4,1.6
3040 DATA 79,XMAS,3,.95,2.65,22,.97
3050 DATA 80,A.DAY,1,.2,.4,10,.2
3060 DATA 80,DOGS,5,1.85,3.7,33,1.85
3070 DATA 80,BIRDS 3,9,3.98,7.5,4,3.98
3080 DATA 80,22C COMMS,2,.44,.9,8,.44
3090 DATA 80,FOLKLORE,5,1.1,2.5,23,1.1
3100 DATA 80,WELFARE,4,.88,.88,7,.88
3110 DATA 80,S.WK,5,1.1,1.1,7,1.1
3120 DATA 80,S.WK M/S,1,.66,.66,7,.66
3130 DATA 80,XMAS,3,1.03,1.03,7,1.03
3140 DATA 80,AIRCRAFT,4,1.67,1.67,7,1.67
3150 END
```

***** UNIVERSAL LOWER-CASE DRIVER ROUTINES by Edwin Paay *****

For some time now, we have been promising to publish a lower-case driver routine for those readers who have installed a lower-case modification in their '80's. Recently, as part of the development of MICRO-80's own lower-case modification, Eddy Paay developed a pair of universal driver routines. Last month we challenged you to guess how he did it - now read on and see if you were correct.

When the TRS-80 was designed, no allowance was made for lower-case letters to be displayed on the screen. Soon the TRS-80 was put to use as a word processor and the fact that the machine doesn't display lower-case was a real disadvantage. After studying the circuits, it will be seen that it is a relatively simple task to provide the TRS-80 with the correct hardware to allow lower-case characters to appear on the display.

The BASIC interpreter however, converts most lower-case characters to upper-case so they won't show up if they are typed from the keyboard. Therefore a lower-case driver routine to bypass a section of the interpreter is required. During development of the MICRO-80 lower-case kit I heard many complaints from people who bought lower-case kits from other sources. Most complaints arose from the fact that these routines would load into high memory, and consequently would clash with other high memory routines such as printer drivers and other utilities. Other complaints concerned incompatibility with disk systems.

When this sort of routine has to be written, two possible directions can be taken:

1. Take the easy way out and write a dozen different driver routines to suit different system configurations and memory sizes, then dump them all on a cassette tape for the user to sort out.

2. Write a routine that suits all '80 systems irrespective of memory size or configuration.

I thought it would be nice to aim for the latter and consequently had to set the following goals:

(a) The routine must be fully relocatable.

(b) It should not load into program RAM before relocation so it won't over-write anything.

(c) It must be able to be used from Disk or Cassette.

(d) Optionally, after initialization it must be possible to exit to DOS for a disk system or to BASIC for a cassette based system. This is because a disk machine would load the lower-case driver from disk before disk BASIC is called up and initialized, whereas a cassette system would load the driver routine from cassette using the BASIC command "SYSTEM" after which BASIC would have to be re-entered.

(e) Let's throw in a flashing cursor and key-debounce for good measure.

(f) A toggle function should be provided to allow the TRS-80 to behave as a standard typewriter with upper-case as shifted characters or in standard TRS-80 mode with lower-case as shifted characters.

Item (a) means that only relative jumps can be used and not CALLS or absolute jumps which contain addresses in their instructions, unless the jump or CALL is common to all configurations. (b) was a tough one. I considered loading the routine into the I/O buffer area but there isn't much space and it might be over-written during keyboard input before it is relocated. Of course, it is possible to instruct the user to load the driver routine before anything else, which means that the routine can be loaded in high memory and then ask where it has to be relocated. But many users are not aware of the actual addresses of other routines which may clash with the lower-case driver. Then, suddenly it struck me that, as part of the lower-case modification, a new RAM chip is added to the screen RAM, converting it to look like any other RAM in the system.

This meant that I could load and execute a routine straight from screen RAM. The maximum memory size is stored in memory, the value of which depends on how the "MEMORY SIZE" question was answered. The lower-case driver can read this value and use it to calculate the highest address allowed for relocation. Finally, (c) and (d) are accomplished by holding down a key during initialization. When the driver routine has finished relocating itself, it scans the keyboard to see if the SHIFT key is down. If so, DOS will be entered, otherwise control is passed to BASIC.

PROGRAM DESCRIPTION.

At the end of this article, there are two source listings, one is a straight driver routine and the other has key-debounce and a flashing cursor built in. As they are both similar, I will describe how the flashing cursor version works. Line 120 sets the cursor outside the area where the program is going to be loaded. The INITialization routine starts at line 1290. Lines 1290 to 1340 calculate the program length. 1360 checks to see if the SHIFT Key is down. If so, use 4049H to find the memory size, otherwise use 4081H. Newdos-80 users can use the HIGHMEM command to force the lower-case driver to relocate to any position in memory. TRSDOS users can use DEBUG to alter address 4049H directly. Line 1420 does the relocating. 1440 and 1450 load the new memory size in place to protect the lower-case driver routine. 1460 makes sure the stack isn't placed within the lower-case driver routine. 1480 defines new video driver routine address. 1515 to 1560 calculate and define new keyboard driver address to provide key-debounce and flashing cursor. 1620 sets delay counter for flashing cursor. 1630 to 1680 - if Floppy Disk Controller present AND the SHIFT key is down, then re-enter DOS or else re-enter BASIC. Think about this for a while. It allows for a cassette based machine with no expansion interface, a cassette machine with expansion interface (this means there is a disk controller present but no disk drive) or a fully-fledged disk system. 1700 to 1770 make sure that all pointers used by BASIC are altered to reflect the new memory size. The initialization routine is left in screen RAM and discarded after use.

This brings us to the actual lower-case driver routine. The start of the video driver routine in ROM has to be by-passed. This was achieved by altering the video driver location pointer in the initialization procedure above. Lines 360 to 490 duplicate a section of ROM but the offending instructions which stop lower-case from being displayed have been removed.

Finally, the keyboard is intercepted each time it is scanned. This is done at line 790. 790 scans the keyboard. If a character is found, (it will be returned in the A register) control is passed to line 610. 610 resets the flashing cursor delay counter. Otherwise, if the delay counter (line 830 - 870) is zero, flash cursor on or off. 880 decrements the delay counter. 500 to 600 toggle the cursor to on if it is off, or off if it is on. 680 to 710 checks if SHIFT zero was pressed. SHIFT zero tells the driver routine to toggle from TRS-80 to typewriter mode and vice versa. Location "FLAG" is used to tell the system what mode it is in. 930 to 1100 scan the keyboard and do a delay loop which is part of the debounce procedure. Line 1120 tests the toggle FLAG to see if upper-case needs to be converted to lower-case. 1160 to 1190 tests to see if character is alphabetic. 1200 to 1270 converts upper to lower-case or vice versa.

ENTERING THE PROGRAMS.

There are two versions of the lower-case driver. They are almost identical except that one is a straight lower-case driver, whereas the other has a key-debounce and flashing cursor routine built in. The program should be typed in and assembled using an assembler. The object code can then be dumped to tape or disk. The routines can be loaded from tape using the "SYSTEM" command; type the "/" followed by the ENTER key to initialize. Disk users can load the routines straight from DOS by typing their filespec; the SHIFT key will have to be held down as soon as the filespec is entered or control will never be handed back to DOS. Type SHIFT/zero to toggle from typewriter mode to TRS-80 mode or back again.

(As stated above, it is necessary to use an editor/assembler to make use of this program. Readers who do not have an editor/assembler can send for the two driver routines on cassette, price $5.00 plus 50c. p&p, or on disk, price $8.50 plus 50c. p&p. - Ed.)

```
00010           ;**********************************
00020           ;* LOWERCASE DRIVER ROUTINE 1.0   *
00030           ;* FOR MICRO-80 LOWERCASE CONV.   *
00040           ;* CREATED  : 25/05/1981          *
00050           ;* BY       : E. R. PAAY          *
00060           ;* FOR      : MICRO-80            *
00080           ;**********************************
00090           ;
00100           ORG     4020H
00110           DEFW    3E00H
00120           ;
```

```
00130           ORG     3C40H
00140           ;
00150 MEMSIZ    EQU     40B1H
00160 KBVEC     EQU     4016H
00170 DISPLY    EQU     033AH
00180 BACKSP    EQU     0008H
00190 VIDVEC    EQU     401EH
00200 VIDSW     EQU     401DH
00210 BASRDY    EQU     06CCH
00220 DOSRDY    EQU     402DH
00230 FDC       EQU     37ECH
00240 DMEMSZ    EQU     4049H
00250 CURPOS    EQU     4020H
00260 SPOINT    EQU     40A0H
00270 SETPTR    EQU     1B6EH
00280 FLAG      EQU     4018H
00290           DEFM    '****** MICRO-80 LOWER CASE DRIVER VERS 1.0, (C) E. R. PAA
Y *****'
00300           DEFM    '**** THE ROUTINE WILL NOW LOAD INTO SCREEN RAM, PLEASE WA
IT ****'
00310           DEFM    '
      '
00320 LCDRV     DEFW    0
00330           LD      L,(IX+3)
00340           LD      H,(IX+4)
00350           JP      C,49AH
00360           LD      A,(IX+5)
00370           OR      A
00380           JR      Z,BYPAS1
00390           LD      (HL),A
00400 BYPAS1    LD      A,C
00410           CP      20H
00420           JP      C,506H
00430           CP      80H
00440           JP      NC,4A6H
00450 DONE      JP      47DH
00460           JR      DONE
00470 KBCHEK    CALL    3E3H
00480           CP      20H
00490           JR      NZ,PROSES
00500           LD      A,(3880H)
00510           AND     1
00520           LD      A,20H
00530           RET     Z
00540           LD      A,(FLAG)
00550           CPL
00560           LD      (FLAG),A
00570           XOR     A
00580           RET
00590 PROSES    LD      C,A
00600           LD      A,(FLAG)
00610           OR      A
00620           LD      A,C
00630           RET     Z
00640           CP      41H
00650           RET     C
00660           CP      80H
00670           RET     NC
00680 REDO      BIT     5,A
00690           JR      Z,LCONV
00700           RES     5,A
00710 FTEST     CP      7BH
00720           JR      NC,REDO
00730           RET
00740 LCONV     SET     5,A
00750           JR      FTEST
00760 EIND      DEFB    0
00770 INIT      LD      HL,INIT
00780           LD      DE,LCDRV
00790           OR      A
00800           SBC     HL,DE
00810           LD      B,H
00820           LD      C,L
00830           LD      HL,EIND
00840           LD      A,(3880H)
```

```
00850           AND     1
00860           JR      NZ,USEDMS
00870           LD      DE,(MEMSIZ)
00880           JR      BYPASO
00890 USEDMS    LD      DE,(DMEMSZ)
00900 BYPASO    LDDR
00910           EX      DE,HL
00920           LD      (MEMSIZ),HL
00930           LD      (DMEMSZ),HL
00940           LD      SP,HL
00950           INC     HL
00960           LD      (VIDVEC),HL
00970           LD      B,H
00980           LD      C,L
00990           LD      HL,KBCHEK
01000           LD      DE,LCDRV
01010           OR      A
01020           SBC     HL,DE
01030           ADD     HL,BC
01040           LD      (KBVEC),HL
01050           LD      H,B
01060           LD      L,C
01070           LD      DE,3E00H
01080           LD      (CURPOS),DE
01090           LD      A,(FDC)
01100           INC     A
01110           CP      2
01120           JR      C,BYPAS2
01130           LD      A,(3880H)
01140           AND     1
01150           JP      NZ,DOSRDY
01160 BYPAS2    LD      DE,32H
01170           OR      A
01180           SBC     HL,DE
01190           LD      (SPOINT),HL
01200           CALL    SETPTR
01210           XOR     A
01220           LD      (FLAG),A
01230           JP      BASRDY
01240           END     INIT
```

```
00010           ;**********************************
00020           ;* LOWERCASE DRIVER ROUTINE 1.0A *
00030           ;* FOR MICRO-80 LOWERCASE CONV.  *
00040           ;* CREATED  : 25/05/1981         *
00050           ;* BY       : E. R. PAAY         *
00060           ;* FOR      : MICRO-80           *
00080           ;* THIS ROUTINE HAS KEY-DEBOUNCE *
00090           ;**********************************
00100           ;
00110           ORG     4020H
00120           DEFW    3E40H
00130           ;
00140           ORG     3C40H
00150           ;
00160 MEMSIZ    EQU     40B1H
00170 KBVEC     EQU     4016H
00180 VIDVEC    EQU     401EH
00190 BASRDY    EQU     06CCH
00200 DOSRDY    EQU     402DH
00210 FDC       EQU     37ECH
00220 DMEMSZ    EQU     4049H
00230 CURPOS    EQU     4020H
00240 SPOINT    EQU     40A0H
00250 SETPTR    EQU     1B6EH
00260 FLAG      EQU     4018H
00270 COUNT     EQU     4019H
00280 DELAY     EQU     0060H
00290 CURSOR    EQU     5FH
00300 CURSOF    EQU     20H
00310 MAXCNT    EQU     400H
```

```
00320 FLAG1   EQU     4022H
00330         DEFM    'MICRO-80 COMBINED LOWERCASE DRIVER AND KEY DEBOUNCE ROUTI
NE 1.0A'
00340         DEFM    '**** THE ROUTINE WILL NOW LOAD INTO SCREEN RAM, PLEASE WA
IT ****'
00350         DEFM    '                  *** BY E.R.PAAY (C) MICRO-80 ***
      '
00360 LCDRV   DEFW    0
00370         LD      L,(IX+3)
00380         LD      H,(IX+4)
00390         JP      C,49AH
00400         LD      A,(IX+5)
00410         OR      A
00420         JR      Z,BYPAS1
00430         LD      (HL),A
00440 BYPAS1  LD      A,C
00450         CP      20H
00460         JP      C,506H
00470         CP      80H
00480         JP      NC,4A6H
00490 DONE    JP      47DH
00500 SETCUR  LD      HL,(CURPOS)
00510         LD      A,(HL)
00520         CP      CURSOF
00530         LD      A,CURSOF
00540         JR      NZ,RESCUR
00550         LD      A,CURSOR
00560         LD      (FLAG1),A
00570 RESCUR  LD      (HL),A
00580         XOR     A
00590         PUSH    AF
00600         JR      FIXCNT
00610 RESCNT  PUSH    AF
00620         XOR     A
00630         LD      (FLAG1),A
00640 FIXCNT  LD      HL,MAXCNT
00650         LD      (COUNT),HL
00660         POP     AF
00670         EXX
00680         CP      20H
00690         JR      NZ,DBOUNS
00700         LD      A,(3880H)
00710         AND     1
00720         LD      A,20H
00730         JR      Z,DBOUNS
00740         LD      A,(FLAG)
00750         CPL
00760         LD      (FLAG),A
00770 CLRA    XOR     A
00780         RET
00790 KBCHEK  CALL    3E3H
00800         EXX
00810         OR      A
00820         JR      NZ,RESCNT
00830         LD      HL,(COUNT)
00840         LD      A,H
00850         OR      L
00860         LD      A,0
00870         JR      Z,SETCUR
00880         DEC     HL
00890         LD      (COUNT),HL
00900         EXX
00910         RET
00920 DBOUNS  EXX
00930         PUSH    AF
00940 LOOP0   LD      B,7
00950         LD      HL,3801H
00960         LD      DE,4036H
00970         XOR     A
00980 LOOP1   OR      (HL)
00990         EX      DE,HL
01000         AND     (HL)
01010         EX      DE,HL
01020         INC     DE
01030         RLC     L
```

```
01040            DJNZ    LOOP1
01050            OR      A
01060            JR      NZ,LOOPO
01070            LD      BC,500
01080            CALL    DELAY
01090            POP     AF
01100            EXX
01110            LD      C,A
01120            LD      A,(FLAG)
01130            OR      A
01140            LD      A,C
01150            RET     Z
01160            CP      41H
01170            RET     C
01180            CP      80H
01190            RET     NC
01200 REDO       BIT     5,A
01210            JR      Z,LCONV
01220            RES     5,A
01230 FTEST      CP      7BH
01240            JR      NC,REDO
01250            RET
01260 LCONV      SET     5,A
01270            JR      FTEST
01280 EIND       DEFB    0
01290 INIT       LD      HL,INIT
01300            LD      DE,LCDRV
01310            OR      A
01320            SBC     HL,DE
01330            LD      B,H
01340            LD      C,L
01350            LD      HL,EIND
01360            LD      A,(3880H)
01370            AND     1
01380            JR      NZ,USEDMS
01390            LD      DE,(MEMSIZ)
01400            JR      BYPASO
01410 USEDMS     LD      DE,(DMEMSZ)
01420 BYPASO     LDDR
01430            EX      DE,HL
01440            LD      (MEMSIZ),HL
01450            LD      (DMEMSZ),HL
01460            LD      SP,HL
01470            INC     HL
01480            LD      (VIDVEC),HL
01490            LD      B,H
01500            LD      C,L
01510            LD      HL,KBCHEK
01520            LD      DE,LCDRV
01530            OR      A
01540            SBC     HL,DE
01550            ADD     HL,BC
01560            LD      (KBVEC),HL
01570            LD      H,B
01580            LD      L,C
01590            LD      DE,3E40H
01600            LD      (CURPOS),DE
01610            LD      DE,MAXCNT
01620            LD      (COUNT),DE
01630            LD      A,(FDC)
01640            INC     A
01650            CP      2
01660            JR      C,BYPAS2
01670            LD      A,(3880H)
01680            AND     1
01690            JP      NZ,DOSRDY
01700 BYPAS2     LD      DE,32H
01710            OR      A
01720            SBC     HL,DE
01730            LD      (SPOINT),HL
01740            CALL    SETPTR
01750            XOR     A
01760            LD      (FLAG),A
01770            JP      BASRDY
01780            END     INIT
```

***** NEXT MONTH'S ISSUE *****

Next month's issue will contain at least the following programs plus the usual features and articles.

** BREAKOUT - L1/4K **

A simulation of the arcade game. See how many inoffensive walls you can pulverise with fire balls.

** INTERCEPT - L1/4K **

This is a Level 1 version of that populat Level 2 game SNAKE, which was published in our first issue and is featured on our free software cassette.

** CHORD PRACTISE - LEVEL 2 **

Those music buffs amongst our readers can use this program to polish up their chord fingering. It was written by a doctor who lives in a small country town, who found that without access to a regular music teacher and having to learn to play an instrument from books, something like this program was needed. So he wrote it; now you can reap the benefit.

** DOG RACE - LEVEL 2 **

It is said that Australians will gamble on two flies crawling up a wall, (which for our overseas readers' information is probably true), well, in this program you can bet on four Scottie dogs racing across the screen just for a change, and give those poor old flies a rest.

** SHARE CHARTING - LEVEL 2 **

This is a data handling program for the input of daily, weekly or monthly share information and could be used by a retailer to chart retail sales or virtually any other requirement for displaying percentage variation in data.

** PSYCHIC MASTER - LEVEL 2 **

This program was written to test the abilities of a person claiming to have psychic abilities. It will test for telepathy and clairvoyance and describes whether your results have been due to chance or true psychic abilities. Now that's a program I call really different!

APPLICATION FOR PUBLICATION OF A PROGRAM IN MICRO-80

Date .....

Tick where appropriate

To **MICRO-80**
Please consider the enclosed program for . . .

(i) Publication in MICRO-80 .....

(ii) Publication on disk or cassette only .....

(iii) Both .....

Name .....

Address .....

.....

.....

.....Postcode.....

*** * * CHECK LIST * * ***

Please ensure that the cassette or disk is clearly marked with your name and address, program name(s), Memory size, Level I, II, System 1 or 2, Edtasm, System, etc. The use of REM statements with your name and address is suggested, in case the program becomes separated from the accompanying literature.

Ensure that you supply adequate instructions, notes on what the program does and how it does it, etc.

For system tapes, the start, end, and entry points, etc.

The changes or improvements that you think may improve it.

Please package securely — padabags are suggested — and enclose stamps or postage if you want your cassette or disk returned.

***** CASSETTE EDITION INDEX *****

The cassette edition of MICRO-80 contains all the software listed each month, on cassette. All cassette subscribers need do is CLOAD and RUN the programs. Level II BASIC programs are recorded on side 1 of the cassette. Level II machine language programs and Level I programs are recorded on side 2. All programs are recorded twice in succession. The rates for a cassette subscription are printed on the inside front cover of each issue of the magazine.

SIDE 1		I.D.	APPROX. START POSITION CTR-41	CTR-80	SYS-80
BONES	L2/4K	B	7 39	5 26	5 26
NORMAL DISTRIBUTION	L2/4K	N	68 81	46 55	46 55
PHILATELIC ADVISER	L2/16K	P	94 167	64 113	64 113
DISASSEMBLER	L2/16K	D	234 302	158 204	158 204
SIDE 2					
CLOCK	L2/m.1	SYSTEM CLOCK	12 22	8 15	8 15
LOWER-CASE DRIVER	L2/m.1	SYSTEM LCD	37 49	25 33	25 33
LOWER-CASE DRIVER WITH DEBOUNCE	L2/m.1	SYSTEM LCDB	67 79	45 53	45 53
SORT DEMONSTRATION	L1/4K	-	96 119	65 80	- -
ARITHMETIC	L1/4K	-	148 190	100 128	- -

TO:
MICRO-80, P.O. BOX 213, GOODWOOD,
SOUTH AUSTRALIA. 5034.

Please RUSH to me the items shown below:

$ enclosed Date .....

..... 12 month subscription to MICRO-80

..... 12 month subscription to MICRO-80, plus the cassette edition

..... The latest issue of MICRO-80 (see inside front cover for prices)

The MICRO-80 PRODUCTS listed below:

DESCRIPTION	PRICE

TOTAL ENCLOSED WITH ORDER

☐ Cheque ☐ Bankcard ☐ Money Order

Bankcard Account Number

Signature ..... Exp. End .....

NAME .....

ADDRESS .....

Postcode .....

A SPECIAL OFFER!

TO SUBSCRIBERS OF MICRO 80

FOR THE FIRST TIME IN AUSTRALIA!
GENUINE TANDY TRS–80 MICRO-COMPUTERS, HARDWARE AND SOFTWARE AT

GREAT DISCOUNT PRICES!!!

HOW TO ORDER –

1. SELECT ITEMS FROM 1980 TANDY CATALOGUE
2. DEDUCT 10% FROM ADVERTISED PRICES
3. POST US YOUR ORDER STATING DESCRIPTION CAT. No AND A CHEQUE OR MONEY ORDER.

WE WILL –

1. ATTEND TO YOUR ORDER WITHIN 7 DAYS *
2. SUPPLY GOODS SELECTED FREIGHT FREE !
3. SEND ADVERTISING REGULARLY TO KEEP YOU INFORMED OF CURRENT SPECIALS !

* *subject to availability*

to **CONQUEST ELECTRONICS** Pty. Ltd.
212 Katoomba St. KATOOMBA 2780

Please supply –

QTY.	CAT NO	DESC.	ADV. PRICE

SUB TOTAL
LESS 10%
FIND CHEQUE FOR **TOTAL**

SEND **FREIGHT FREE** TO

NAME .....

ADDRESS .....

..... P/ Code .....

TANDY ELECTRONICS DEALER

CONQUEST ELECTRONICS PTY. LTD.

212 KATOOMBA ST KATOOMBA N.S.W. 2780 PHONE (047) 82 2491

MICRO-80

LEVEL II ROM REFERENCE MANUAL

by Edwin Paay

Published by MICRO-80 PRODUCTS

Written by Eddy Paay, the LEVEL II ROM REFERENCE MANUAL is the most complete explanation of the Level II BASIC interpreter ever published.

Part 1 lists all the useful and usable ROM routines, describes their functions explains how to use them in your own machine language programs and notes the effect of each on the various Z 80 registers.

Part 1 also details the contents of system RAM and shows you how to intercept BASIC routines as they pass through system RAM. With this knowledge, you can add your own commands to BASIC, for instance, or position BASIC programs in high memory—the only restriction is your own imagination!

Part 2 gives detailed explanations of the processes used for arithmetical calculations, logical operations, data movements, etc. It also describes the various formats used for BASIC, SYSTEM and EDITOR/ASSEMBLER tapes. Each section is illustrated by sample programs which show you how you can use the ROM routines to speed up your machine language programs and reduce the amount of code you need to write.

The LEVEL II ROM REFERENCE MANUAL is intended to be used by machine language programmers. It assumes a basic understanding of the Z 80 instruction set and some experience of Assembly Language programming. But BASIC programmers too will benefit from reading it. They will gain a much better insight into the functioning of the interpreter which should help them to write faster, more concise BASIC programs.

MICRO-80